Theories and Proof Systems for PSPACE and the EXP-Time Hierarchy

by

Alan Ramsay Skelley

A thesis submitted in conformity with the requirements
for the degree of Doctor of Philosophy
Graduate Department of Computer Science
University of Toronto

Abstract

Theories and Proof Systems for PSPACE and the EXP-Time Hierarchy

Alan Ramsay Skelley

Doctor of Philosophy

Graduate Department of Computer Science

University of Toronto

2006

This dissertation concerns theories of bounded arithmetic and propositional proof systems associated with PSPACE and classes from the exponential-time hierarchy.

The second-order viewpoint of Zambella and Cook associates second-order theories of bounded arithmetic with various complexity classes by studying the definable functions of strings, rather than numbers. This approach simplifies presentation of the theories and their propositional translations, and furthermore is applicable to complexity classes that previously had no corresponding theories. We adapt this viewpoint to large complexity classes from the exponential-time hierarchy by adding a third sort, intended to represent exponentially long strings ("superstrings"), and capable of coding, for example, the computation of an exponential-time Turing machine. Specifically, our main theories W^i_1 and TW^i_1 are associated with $\mathrm{PSPACE}^{\Sigma^p_{i-1}}$ and $\mathrm{EXP}^{\Sigma^p_{i-1}}$, respectively. We also develop a model for computation in this third-order setting including a function calculus, and define third-order analogues of ordinary complexity classes. We then obtain recursion-theoretic characterizations of our function classes for FP, FPSPACE and FEXP. We use our characterization of FPSPACE as the basis for an open theory for PSPACE that is a conservative extension of a weak PSPACE theory HW^0_1.

Next we present strong propositional proof systems QBP_i, which are based on the Boolean program proof system BPLK but additionally with quantifiers on function symbols. We exhibit a translation of theorems of W^i_1 into polynomial-sized proofs in QBP_i.

Acknowledgements

Without being excessively mushy, let me begin by saying that my parents have really exceeded all expectations. NSERC once again came through for me with PGSB-208264-2000, while a Walter C. Sumner memorial fellowship was also much appreciated.

On the academic front, I count myself very fortunate to have had Stephen Cook as supervisor. Not only is he kind, generous, infinitely patient, and far more hardworking than anyone has any right to expect, but he is also essentially all-knowing. My supervisory committee was rounded out by: Alasdair Urquhart, who always seems to have read something relevant; Charles Rackoff, who always has a different perspective; and Toniann Pitassi, who always cuts through the notation to get to the real point. Many thanks also to my external appraiser, Sam Buss.

My colleagues, office mates and friends provided plenty of assistance and distraction. Any attempt to enumerate them would be futile and ill-advised; but chief among them is my wife, Kleoni Ioannidou, who fits into all three categories and more.

Contents

Chapter 1

Introduction

We often argue that a particular mathematical concept is important if it is natural, which means that it surfaces in many places with different origins and definitions, and robust, such that a variety of disparate formulations of it end up being equivalent or at least closely related. Likewise, the applicability, maturity, and importance of a body of results are greater when that field is found to have a strong connection to another. Three areas of study intricately connected in such a useful way are computational complexity, the proof theory of arithmetic and propositional proof complexity.

Computational complexity is the study of computation and the resources required to perform it. A staggering number of different kinds of computation all fall into the domain of this field. It has practical aspects, directly impacting how real computations are done by real computers, and yet seemingly fundamental, easily explained problems remain unsolved despite a good deal of effort. A particularly glaring example is the famous P vs NP problem, which asks if those two classes of problems are equal. Starting from the NP-completeness results of Cook [24] the pressure mounted with no relief, leading even to detailed, formal analysis of known proof techniques and why they are all ineffectual at tackling such problems [51]. Many complexity classes are studied and conjectures about separations and hierarchies abound, yet results are elusive.

A different way of studying computational complexity is indirectly through logic, and in particular, bounded arithmetic. Many connections between the fields are known: among them, that complexity classes can be characterized as those sets or functions definable in certain theories, and that predicates or functions from certain complexity classes can be used to define new logics in various ways. Results about either area can have implications for the other.

Bounded arithmetic and propositional proof systems are related in several ways: due to Cook [25] and others, there are translations from formulas of bounded arithmetic to polynomial-sized families of propositional or quantified propositional formulas which additionally have very interesting properties relating the theories and the proof systems, and also have complexity implications. Another connection is that a theory's ability to prove different kinds of consistency of related propositional proof systems has a bearing on its power relative to other theories, and the relative complexity of proofs in the proof systems.

Finally, the full circle back to computational complexity is completed with the work of Cook and Reckhow in [20] and [28]. They show that NP=co-NP if and only if there exists a polynomially bounded proof system, and additionally introduce many of the important definitions in the area such as those of proof systems, polynomial simulations, and so on. These results, and others concerning the complexity of witnessing proofs of quantified propositional formulas, drive the study of propositional proof complexity and the search for lower bounds on propositional proof systems. Fine examples are the superpolynomial lower bounds for resolution, due to Haken [32] and bounded depth Frege systems, due to Ajtai [2]. For many seemingly stronger systems, however, no such results are known.

While it is well understood that the study of the intrinsic hardness of computation is crucially relevant to many fields of human endeavour, it is perhaps less well appreciated that propositional proof complexity and bounded arithmetic collectively reach for a similar goal: the study of the intrinsic hardness of reasoning. Mathematics has long been contributing to the benefit of mankind behind the scenes while the benefits of computation are public and evident; but just as the limitations of computation have become apparent, so now are the limitations of reasoning. As the use of automated theorem proving and other formal or mechanical forms of reasoning become more necessary to handle the vast amounts of data that mankind processes, to verify our increasingly complicated machinery and computer systems, and to assist in expanding the frontiers of mathematics, it will be necessary to understand more about reasoning itself.

1.1 Overview of Dissertation

In this dissertation we present bounded arithmetic theories and propositional proof systems corresponding to large complexity classes, including polynomial space (PSPACE), (polynomially-)exponential time (EXP) and the levels of the exponential-time hierarchy. Buss originally defined second-order (i.e., two-sorted) theories U_2^1 and V_2^1 so that strong (PSPACE and EXP) number functions could be defined by reasoning about exponential-length computations. Then using the second sort in a completely different way, Ignjatovic, Zambella, Cook and more recent authors have used second-order theories to great effect to capture the string-based computation of Turing machines more simply than with previous one-sorted theories of bounded arithmetic. Furthermore, this second-order "viewpoint" has allowed the development of theories for some very small complexity classes previously not captured with one-sorted theories due to their inherent coarseness. We combine the two approaches and extend the second-order viewpoint to higher complexity classes in a natural way by adding a third sort to represent exponentially large objects such as computations from these strong classes, outputs of unbounded exponential-time functions, or even oracles. We thereby obtain the benefits of the second-order viewpoint (simpler presentation of theories, more intuitive relationship to string-based complexity) while retaining the ability to reason about exponential-sized objects.

Another contribution of this dissertation is to define a calculus of functions that operate on these three sorts of objects. From a complexity standpoint for the classes we are interested in, the objects are the usual binary strings, always of polynomial length, represented by finite sets of natural numbers; numbers, which are to be thought of as short inputs and presented in unary for the purposes of resource bounds; and finally, superstrings: exponential-length strings indexed by standard binary strings, and not counted in any resource bounds. This function calculus is very nicely suited for expressing the computational objects reasoned about by our third-order theories of bounded arithmetic; additionally, it is useful for discussing with one unified notation both polynomially-bounded functions and more exotic functions dealing with exponential-sized inputs and outputs. We define complexity classes of functions and predicates, and in each case the string functions or string predicates in these classes constitute exactly the corresponding complexity class of polynomially-bounded string functions or languages. Here the third-order viewpoint pays off again, as it allows computation to involve exponential-length objects, yet keeps them separate from "ordinary" inputs and thus takes care of the problem of composability of functions.

Finally, we demonstrate translations of certain theorems of some of these theories into polynomial-size families of propositional proofs. These translations are Cook-style in the sense that the focus (generalized computation model notwithstanding) is on definable functions of strings, and the propositional variables translate free string variables; this is as opposed to the Paris-Wilkie translations in which propositional variables translate a higher-order predicate added to the language. The first proof system we use is BPLK, which is a standard sequent calculus modified to allow the use of Boolean programs. These are a notation for specifying Boolean functions concisely, and formulas in this enhanced language are PSPACE-complete to evaluate; this is somewhat analogous to the existing translation of Σ^b_∞ theorems of U^1_2 into G due to Krajíček and Takeuti [42], as the quantified Boolean formulas in G are also PSPACE-complete to evaluate. We also define quantified versions of BPLK for translating theorems of the stronger theories, which does not seem to have an existing analogue in the literature.

The remainder of the dissertation is organized as follows: We begin in Chapter 2 by surveying some relevant literature and presenting some important definitions and results therefrom. This chapter includes some material and definitions that are fundamental to this dissertation, but it also describes other results that, while relevant, are not strictly necessary for this dissertation.

Chapter 3 addresses the language of third-order arithmetic, as well as third-order computation, including in particular recursion-theoretic characterizations of some complexity classes from the third-order point of view. This is new; although some authors have discussed higher-order computation before, it hasn't been fully addressed, particularly not with respect to defining functions in full generality as we aim to do. Previous higher-order theories for our classes were number-based, and thus did not follow the newer Zambella-Cook framework of string-based theories.

Chapter 4 introduces the third-order axiom schemes and theories we will discuss: W_1^i and TW_1^i, analogues of U_2^i and V_2^i (see below for the intended complexity classes), as well as HW_1^0, a weaker PSPACE theory based on a recursion scheme, and TTW_1^0, an exponential-time theory which is to W_1^0 and W_1^1 as TV^0 is to V^0 and V^1. In this chapter we also prove some basic results such as a third-order Parikh's theorem, and discuss replacement schemes.

Chapter 5 concerns definability in the theories. This includes our definition of third-order definability, which covers the most general case of arguments and function value of any sort. These were not obvious, especially in the case of superstring-valued functions, which are problematic as our third-order variables are unbounded. We then prove that for $i > 0$, $(\mathrm{FPSPACE}^{(\Sigma_{i-1}^{exp})^\diamond})^+$ and $(\mathrm{FEXP}^{(\Sigma_{i-1}^{exp})^\diamond})^+$ are $\Sigma_i^{\mathcal{B}}$-definable in respectively W_1^i and TW_1^i; restricted to strings, these classes are exactly the usual complexity classes $\mathrm{FPSPACE}^{\Sigma_{i-1}^p}$ and $\mathrm{FEXP}^{\Sigma_{i-1}^p}$. We also prove definability results for HW_1^0, TTW_1^0 and

$\widehat{W_1^1}$. Although these results have lower-order analogues, they are nevertheless somewhat new; in particular, our theories are like Buss's original U_2^1 and V_2^1 in that they are "unbounded domain"; later results about U_2^i and V_2^i and the exponential-time hierarchy pertain to Razborov's "bounded domain" versions, which sidestep the problems of unbounded higher-order objects.

Chapter 6 is somewhat of a side-bar into a universal conservative extension of HW_1^0, a candidate for a minimal PSPACE theory. This was developed with a view to using the Cook-Thapen [22] argument to use this open theory to show that HW_1^0 does not prove the replacement scheme, subject to a complexity assumption.

Chapter 7 contains witnessing theorems for W_1^i, TW_1^i, TTW_1^0 and HW_1^0 (to match the results about definable functions in these theories).

Chapter 8 introduces quantified propositional proof systems based on BPLK, and then uses these systems to translate $\Sigma_i^{\mathcal{B}}$-theorems of W_1^i.

Finally, in chapter 9 we conclude with a discussion of some open problems.

Chapter 2

Preliminaries and Related Work

In this chapter we survey various results from the literature hinted at in the introduction; at the same time we also reprise some definitions and results that are important for the dissertation. We intentionally omit from our focus some of the weaker systems of bounded arithmetic and propositional proof systems about which some lower bounds are known and concentrate instead on stronger systems and theories about which good bounds or separations are only conjectured. The survey is organized as follows: In section 2.1 we introduce several systems of bounded arithmetic and results concerning them and relating them to complexity theory. In section 2.2 we present some relevant propositional proof systems and their complexity-theoretic ramifications. Finally, section 2.3 contains a discussion of results relating bounded arithmetic and propositional proof systems.

2.1 Bounded Arithmetic and Complexity

The study of bounded arithmetic was initiated in 1971 by Parikh with his system $I\Delta_0$, similar to Peano Arithmetic, but with the important restriction of the induction scheme to Δ_0 formulas: those whose quantifiers are bounded, i.e. of the form $\forall x(x \leq t \to \phi(x))$ or $\exists x(x \leq t \wedge \phi(x))$ for some term t with no occurrences of x. An important consequence of this restriction is given by Parikh's theorem, which states that any function which can be proved total in $I\Delta_0$ can be bounded by a term in the language:

Theorem 2.1.1 (Parikh, 1971). *Assume that $\theta(\overline{a}, b)$ is a Δ_0 formula and that*

$$I\Delta_0 \vdash \forall\overline{x}\exists y\theta(\overline{x}, y).$$

Then there is a term $t(\overline{x})$ such that

$$I\Delta_0 \vdash \forall\overline{x}\exists y < t(\overline{x}), \theta(\overline{x}, y).$$

This theorem implies that $I\Delta_0$ cannot prove theorems requiring exponentiation, or even the existence of numbers whose length is polynomial in the length of input parameters. This rules out reasoning about any computations using more than linear time, many logical constructions such as substitution and polynomial-length sequences, and so on.

The first logical theory designed to reason about all "feasible," i.e. polynomial-time concepts, and only those concepts, was Cook's PV [25]. This is an equational theory which has function symbols for every polynomial-time function, and is defined with the help of Cobham's earlier characterization of polynomial-time functions as the closure of a certain set of initial functions under composition and limited recursion on notation. A fundamental property of PV is its connection to the propositional proof system EF, which will be discussed in section 2.3.1. Later first-order theories IPV and CPV [23], the former intuitionistic and the latter classical, have a more expressive language allowing interesting properties of graph theory and combinatorics to be stated, yet are conservative over PV, which is to say that every statement in the language of PV which is provable in IPV or CPV is provable in PV. Krajíček, Pudlák and Takeuti [41] defined a hierarchy of theories PV_i based on PV, whose lowest member, PV_1, is also referred to as QPV. Using a similar framework to that of PV, Dowd [29] defined an open theory PSA for reasoning about PSPACE number functions and gave a translation of theorems of that theory into a quantified propositional calculus.

Following Cook's work on PV, to remedy the deficiency in $I\Delta_0$ Paris and Wilkie tried adding function symbols with faster growth rates such as $\omega_1(x) := x^{|x|}$, along with appropriate defining axioms, to the language. The addition of ω_1 in particular results in a very interesting theory $I\Delta_0 + \Omega_1$ with many implications for and connections to complexity theory and propositional proof systems; Buss's seminal dissertation [3] is the start of a long line of research on subtheories of this important theory, and stronger theories (e.g., U_2^1) defined by allowing reasoning about exponential-size objects, but still with the focus being on definable functions of numbers.

Starting with Ignjatovic [34] and Zambella [58],[59], many authors have investigated second-order (two-sorted) arithmetic as a means to obtain theories with simpler axiomatizations; as computation is typically defined on strings, this makes bootstrapping the theory easier, and additionally allows capturing very weak complexity classes that may not even contain functions such as multiplication. Theories in this vein include V_1^1 for polynomial time [50] (later streamlined by Cook to V^1), and now many theories for classes such as V^0 for AC^0 [14], VTC^0 for TC^0 [44] [45], V-Krom for NL [17], VNC^1 for NC^1 [18] and so on.

Also relevant to this dissertation's focus on the exponential-time hierarchy and higher-order bounded arithmetic is work by Clote and Takeuti [12] in which the authors present order $N+1$ theories for N-fold exponential time (a stack of N 2's topped by a polynomial). The authors separate the time contribution from each variable in a multivariate function as a way to address composability of functions, which can be seen as a coarser-grained version of our third-order computation. (Necessarily so, in order to address such large complexity classes).

In the remainder of this section we include some of the most relevant definitions and statements of important theorems concerning the theories mentioned above, and discuss some of the more interesting consequences.

2.1.1 Definitions

Buss [3] introduced the first-order theories S_2^i and T_2^i, which we shall describe shortly. First, consider the following hierarchy of classes of formulas $g\Sigma_i^b$ and $g\Sigma_i^b$, the definition of which is a slight adaptation of the definition in [36]. They are defined by counting alternations of bounded (first-order) quantifiers (this is what the 'b' superscript refers to) and ignoring sharply bounded quantifiers. Buss called these classes Σ_i^b and Π_i^b, but following the notation of [19], we prepend a "g" to emphasize that these formulas have i alternations in the more general sense (i.e., not requiring strict quantifier syntax). This definition excludes mention of what language, specifically, is used, although it must include '$<$'. This is a failure of the notation in common use, but in practise when discussing a particular theory, the language is clear from the context.

Definition 2.1.2 (Buss, 1986). *$g\Sigma_i^b$ and $g\Pi_i^b$ are the smallest classes of formulas satisfying the following:*

1. *$g\Sigma_0^b = g\Pi_0^b$ are the sharply bounded formulas (meaning that all quantifiers are of the form $(Qx < |t|), Q \in \{\forall, \exists\}$ for some term t).*
2. *If ϕ is $g\Sigma_i^b$ or $g\Pi_i^b$ then it is also $g\Sigma_j^b$ and $g\Pi_j^b$ for all $j > i$.*
3. *If $\phi(x)$ is $g\Sigma_i^b$ then $\forall x < t(x)\phi(x)$ is $g\Pi_{i+1}^b$.*
4. *If $\phi(x)$ is $g\Pi_i^b$ then $\exists x < t(x)\phi(x)$ is $g\Sigma_{i+1}^b$.*
5. *If ϕ is $g\Sigma_i^b$ ($g\Pi_i^b$) then $\neg\phi$ is $g\Pi_i^b$ ($g\Sigma_i^b$ respectively).*
6. *$g\Sigma_i^b$ and $g\Pi_i^b$ are closed under $\vee$ and $\wedge$.*
7. *$g\Sigma_i^b$ ($g\Pi_i^b$) is closed under bounded existential (universal) quantification and sharply bounded quantification.*

Now with these classes in mind, S_2^i and T_2^i are theories over the language $\mathcal{L}_A^1$ consisting of the language $\mathcal{L}_{PA} := \{0, 1, +, \cdot, <, =\}$ of PA with the addition of $\{\lfloor \frac{x}{2} \rfloor, |x|, x\#y\}$, where it is intended that $x\#y = 2^{|x||y|}$. Both theories contain BASIC, a set of 32 open axioms expressing properties of the symbols in the language, and in addition T_2^i has the scheme $g\Sigma_i^b$-IND, while S_2^i has instead the scheme $g\Sigma_i^b$-PIND. We now define these induction schemes in general:

Definition 2.1.3. *Let Φ be a class of formulas. Then Φ-IND consists of the scheme*

$$\phi(0) \wedge \forall x(\phi(x) \rightarrow \phi(x+1)) \rightarrow \forall x \phi(x)$$

for all $\phi \in \Phi$, while Φ-PIND consists of

$$\phi(0) \wedge \forall x(\phi(\lfloor \frac{x}{2} \rfloor) \rightarrow \phi(x)) \rightarrow \forall x \phi(x)$$

(and in this case the language must include '$\lfloor \frac{x}{2} \rfloor$').

The class of induction formulas of S_2^i and T_2^i is often called "Σ_i^b with smash" to emphasize the inclusion of '#'; we would call them "$g\Sigma_i^b$ with smash". The "2" subscript in the names of the theories refers to the presence of the smash function in the language. Other possibilities for this subscript include "1", meaning that no smash function is present, or $i > 2$, meaning that the function $x\#_i y := 2^{|x|\#_{i-1}|y|}$ is present; in these case the set of open axioms defining the language would of course be different.

Clearly $S_2^i \subseteq T_2^i$, and it can be shown that $T_2^i \subseteq S_2^{i+1}$. Buss also defined second-order theories U_2^1 and V_2^1, where the second-order objects are either function variables (expressing a polynomially bounded function from numbers to numbers) or (number) predicate variables. He first gives a definition analogous to 2.1.2 for bounded second-order formulas, where $g\Sigma_i^{1,b}$ formulas (with smash) are classified by counting the alternations of (unbounded) second-order quantifiers and ignoring bounded first-order quantifiers. (The superscript "$1, b$" means that unbounded second-order quantifiers are counted and bounded first-order quantifiers ignored.) U_2^1 (respectively V_2^1) is the theory composed of the BASIC axioms, $\Sigma_0^{1,b}$-Comprehension, which postulates the existence of second-order objects equivalent to given $\Sigma_0^{1,b}$ predicates of one variable, and $g\Sigma_1^{1,b}$-PIND (respectively, $g\Sigma_1^{1,b}$-IND).

Actually Buss defines several versions of each theory, varying in whether function or predicate variables, or both, are allowed in formulas, comprehension, or induction. In all cases, though, the $\Sigma_i^{1,b}$-definable functions (see below) of all the versions of U_2^i (or V_1^i) are the same, and most authors, e.g. [36], consider only the now most standard versions $\tilde{U}_2^i(BD)$ and $\tilde{V}_2^i(BD)$. The BD stands for **bounded domain** and specifies that the second-order variables are tagged with bounds, and the class of formulas allowed in the induction is defined by counting bounded second-order quantifiers. Without these tags, the RSUV isomorphisms (see below) would not hold.

Recent literature on higher-order theories has adopted the alternative notation Σ_i^B for (strict quantifier syntax) $\Sigma_i^{1,b}$ with bounded second-order variables, and we follow suit. The superscript 'B' is intended to mean that it is alternations of bounded second-order quantifiers that are being counted. Strictly speaking this notation is not applicable to Buss's original theories wherein second-order variables are unbounded; however, when reasoning about functions of numbers, second-order variables are all implicitly bounded in the sense that the portion of a second-order variable relevant to the truth-value of a $\Sigma_i^{1,b}$-formula is bounded due to the number quantifiers all being bounded. In the present context, then, we consider Σ_i^B and $\Sigma_i^{1,b}$ to be interchangeable.

A crucial definition is that of definability:

Definition 2.1.4. *Let Φ be a class of formulas, T be a theory of bounded arithmetic and $f : \mathbb{N}^k \to \mathbb{N}$ a function. Then f is Φ-definable in T iff there exists a formula $D_f(\overline{x}, y) \in \Phi$ such that*

$$T \vdash \forall \overline{x} \exists ! y D_f(\overline{x}, y),$$

and $D_f(\overline{x}, f(\overline{x}))$ is true in the standard model.

A function is sometimes called weakly definable to assert only that the theory proves the existence of the y satisfying $D_f(\overline{x}, y)$, but not necessarily that it is unique. In this case it would be also necessary to require explicitly that D_f be the graph of f, and so either there is a unique y satisfying $D_f(\overline{x}, y)$ (but not provably so), or f is a multifunction.

As the final point of this subsection, we outline the theories V^i, which form the foundation for our theories in this dissertation. The language $\mathcal{L}_A^2$ of these theories consists of the smash-free set $\{0, 1, +, \cdot, |\cdot|, \in_2, =_1, =_2\}$ of nonlogical symbols and additionally includes variables of two sorts for numbers and finite sets of numbers (strings). The set 2-BASIC of axioms defining properties of this language is:

B1. $x + 1 \neq 0$	B8. $(x \leq y \wedge y \leq x) \supset x = y$
B2. $x + 1 = y + 1 \supset x = y$	B9. $0 + 1 = 1$
B3. $x + 0 = x$	B10. $0 \leq x$
B4. $x + (y + 1) = (x + y) + 1$	B11. $x \leq y \wedge y \leq z \supset x \leq z$
B5. $x \cdot 0 = 0$	B12. $x \leq y \vee y \leq x$
B6. $x \cdot (y + 1) = (x \cdot y) + x$	B13. $x \leq y \leftrightarrow x < y + 1$
B7. $x \leq x + y$	B14. $x \neq 0 \supset \exists y \leq x(y + 1 = x)$
L1. $X(y) \supset y < \lvert X \rvert$	L2 $y + 1 = \lvert X \rvert \supset X(y)$

SE. $X = Y \leftrightarrow [|X| = |Y| \wedge \forall i < |X|(X(i) \leftrightarrow Y(i))]$

Now the theory V^i consists of the universal closure of these axioms plus the Σ_i^B-COMP comprehension scheme, which is

$$(\exists Y \leq t(\overline{x}, \overline{X}))(\forall z \leq a)[\phi(\overline{x}, \overline{X}, z) \leftrightarrow Y(z)]$$

for every $\phi \in \Sigma_i^B$. V^i proves Σ_i^B-IND (see Definition 2.1.3) by means of the comprehension and a minimization scheme on sets. V^0 is a conservative extension of $I\Delta_0$.

2.1.2 Definability and Witnessing Theorems

In this section we list some of the important results connecting theories of bounded arithmetic and complexity classes through definability of functions.

The main results of Buss [3] are as follows: Firstly, that the strongly Σ_i^b-definable functions of S_2^i are exactly those computable in polynomial time with an oracle for a

Σ^p_{i-1} predicate, i.e. functions from a functional version of the well known polynomial-time hierarchy; this function class is referred to as $\square^p_i$:=$\mathrm{FP}^{\Sigma^p_{i-1}}$. Furthermore, if S^1_2 proves that a predicate is in NP∩co-NP, then it is in fact in P. He also shows how to relativize S^i_2 by adding a free second-order variable, and that an analogous definability result connects these theories to computations with an oracle. Secondly, he shows that S^1_2(PV), which is S^1_2 extended by the language of PV and axioms defining all its function symbols, is conservative over PV. Finally, Buss shows that that the strongly $\Sigma^{1,b}_1$-definable functions of U^1_2 and V^1_2 are those computable in polynomial space, and those computable in exponential time, respectively. These latter results for second order theories are extended by Buss, Krajíček and Takeuti [5] to $\mathrm{EXP}^{\Sigma^p_{i-1}}$[wit,poly] **multifunctions** for U^i_2 and $\square^{exp}_i$:=$\mathrm{EXP}^{\Sigma^p_{i-1}}$ functions in V^i_2.

The many analogues between first- and second-order theories are seen to be part of a pattern formalized in the RSUV isomorphism of Takeuti [57] and Razborov [49]. This isomorphism states that there are translations from first- to second-order formulas and vice versa such that certain pairs of first- and second-order theories have the same theorems modulo the translation. Such pairs include (R^i_{j+1},U^i_j) and (S^i_{j+1},V^i_j) for $i, j > 0$. Therefore as one might expect, in the second-order viewpoint the Σ^B_i-definable functions of V^i are precisely the $\square^p_i$ (i.e., $\mathrm{P}^{\Sigma^p_{i-1}}$) functions for $i > 0$ (and additionally FAC^0 for $i = 0$). The theories TV^i defined to have exponentially longer induction than V^i similarly correspond to T^i_2.

Later results have added to what is known about definability in these theories. Of particular interest is the fact from [4] that the definable functions of T^1_2 are exactly those expressible as the composition of a PLS problem and a projection, where PLS is Papadimitriou's NP search class of polynomial local search problems. Chiari and Krajíček have extended this result to characterize the Σ^b_2 and Σ^b_3 definable multifunctions in T^2_2 as oracle PLS problems and suggest that a more complete understanding of these and related definabilities will be useful for proving non-conservation results. Another important example, which will figure prominently in the next subsection, is the Krajíček-Pudlák-Takeuti (KPT) witnessing theorem [41]:

Theorem 2.1.5 (Krajíček, Pudlák and Takeuti, 1991)**.** *Let $i \geq 1$ and assume that $\phi(a, x, y)$ is an $\exists\Pi_i^b$-formula. Suppose*

$$T_2^i \vdash \exists x \forall y, \phi(a, x, y)$$

Then there are $\square_{i+1}^p$-functions $f_1(a), f_2(a, b_1), ..., f_k(a, b_1, ..., b_{k-1})$ with all free variables shown such that T_2^i proves

$$\phi(a, f_1(a), b_1) \vee \phi(a, f_2(a, b_1), b_2) \vee ... \vee \phi(a, f_k(a, b_1, ..., b_{k-1}), b_k)$$

This is also true for PV_{i+1} in place of T_2^i and for PV_1 if $i = 0$.

2.1.3 Relating the Collapse of Theories with the Collapse of Complexity Classes

Since results are known characterizing fairly precisely the definable functions of many theories, it is reasonable to expect some relation between questions of theories coinciding versus questions of complexity classes coinciding. This is certainly the case of the $S_2 = T_2$ hierarchy under discussion, which will serve as a good example. Something to note at the start is a nice feature of the theories S_2^i and T_2^i, namely that each of them is finitely axiomatizable [40]; therefore, the S_2 hierarchy collapses iff S_2 itself is finitely axiomatizable.

Now, if it were the case not only that the polynomial hierarchy collapsed, but also that this collapse was uniform enough that S_2 could prove it, then the S_2 hierarchy would also collapse. This is so intuitively because some sufficiently high level of S_2 would be strong enough to prove all the induction axioms of S_2, by proving them equivalent due to the PH collapsing to induction axioms of lower quantifier complexity. There is still however the possibility that the PH could collapse but that the proof of that fact might not be formalizable in S_2, in which case the S_2 hierarchy might still be strict. This type of relationship seems to be typical of theories and the complexity classes of functions definable in them; for another example see Cook [26].

In the other direction, the KPT witnessing theorem stated above implies that if the S_2 hierarchy collapses then so does the PH. Buss [7] and Zambella [58] independently strengthen this result by showing that the collapse of the PH would in fact be provable in S_2.

A general pattern is that the collapse of complexity classes seems to be related most closely to the collapse of particular fragments of related theories. In many cases, the status of other fragments of the theories may have different or unknown implications. For example, the collapse of the universal fragments of the theories S_2^i does not obviously imply the collapse of the entire theories (and thus of the PH). Another example is that although we know that $S_2^1(\mathrm{PV})$ is conservative over PV, as is QPV, the KPT witnessing theorem just discussed tells us that if S_2^1 is conservative over QPV, then the PH collapses. Finally, it is not known how the potential equality of PSPACE and PH may be related to the question of conservativity of U_2^1 over S_2, although it is plausible that some relation may hold. Certainly there are many unsolved problems of this kind which are meritorious of further attention.

2.1.4 Candidates for Separation

The standard candidate for separating a theory from one containing it would be the consistency of the smaller theory. However, Paris and Wilkie [47] show that even S_2 augmented with an axiom stating the totality of exponentiation does not prove the consistency of the induction-free Robinson's Arithmetic Q. Not even $\mathrm{BdCon}(S_2^1)$, a restricted consistency statement asserting only that the bounded fragment of S_2^1 is consistent, can be proved in S_2 [48]. More natural candidates, then, would be theorems of mathematics whose proofs require reasoning about concepts which are not in the corresponding complexity class of definable functions of the weaker theory; however, actually finding these seems to be difficult. The most natural candidates appear to be statements of consistency of related propositional proof systems, which will be discussed in section 2.3.2.

2.2 Propositional Proof Systems and Complexity

In this section we discuss some of the many connections between propositional proof systems, which we first formally define, and complexity. The first connection is visible even as the definitions are presented; namely, that when formulated in a Gentzen sequent style, many known propositional proof systems can be seen to be very similar, with the only difference between them being the computational power of what can be written at each line of the proof (or alternatively, what is allowed in the **cut** rule). Examples are Boolean formulas in Frege systems, single literals in resolution, Boolean circuits in extended Frege systems. Another example is the system G, which is a sequent-based system where formulas in the sequents are quantified Boolean formulas (QBFs). These formulas have propositional variables and also propositional quantifiers. In this case, then, since evaluating QBFs is PSPACE-complete, the computational power which can be harnessed in sequents is PSPACE. We can restrict G to G_i by restricting the number of alternations of quantifiers allowed in the formulas, and the reasoning power is then that of Σ_i^p predicates.

2.2.1 Preliminaries

Propositional Proof Systems

Definition 2.2.1. *A* proof system *P for a set S is a surjective polynomial-time computable function $P : \Sigma^* \to S$ for some alphabet Σ.*

We are interested in proof systems both for TAUT, the set of (quantifier-free) propositional tautologies, and for TAUT_i, the set of quantified propositional tautologies from

$\Sigma_i^q \bigcup \Pi_i^q$, to be defined below. A P-proof of a tautology τ is a string π such that $P(\pi) = \tau$. We denote by $|\pi|$ the number of symbols in π. We have the following important definition which allows us to compare the power of proof systems:

Definition 2.2.2. *If P and Q are proof systems, we say that P* polynomially simulates *(p-simulates) Q and write $Q \leq_p P$ if there is a polynomial-time computable function g such that for every string x, $P(g(x)) = Q(x)$.*

Though proof systems need not be of this form, proofs in many of the systems commonly studied are sequences of lines, where each line is a valid statement in some language. Such systems then have a treelike subsystem, wherein each line may be used only once as a hypothesis.

LK and Quantified Propositional Logic

A popular proof system is Gentzen's sequent system LK. LK is actually a proof system for predicate logic but we shall consider only the propositional fragment. Each line of an LK-proof is a sequent, a string of the form $\Gamma \longrightarrow \Delta$, where Γ and Δ are possibly empty finite sequences of propositional formulas. A sequent is satisfied if and only if either one of the formulas on the left (the *antecedent*) is falsified, or one of the formulas on the right (the *succedent*) is satisfied. Each sequent in a proof is either an initial sequent of the form $0 \longrightarrow$, $\longrightarrow 1$ or $a \longrightarrow a$ for an atom a, or it is derived from previous ones (its hypotheses) via one of the following inference rules (this set is the same as in [15], which is a slight modification of the ones in [36]):

weakening:

$$\textbf{left} \quad \frac{\Gamma \longrightarrow \Delta}{A, \Gamma \longrightarrow \Delta} \qquad \text{and} \qquad \textbf{right} \quad \frac{\Gamma \longrightarrow \Delta}{\Gamma \longrightarrow \Delta, A}$$

exchange:

$$\textbf{left}\quad \frac{\Gamma_1, A, B, \Gamma_2 \longrightarrow \Delta}{\Gamma_1, B, A, \Gamma_2 \longrightarrow \Delta} \qquad \text{and} \qquad \textbf{right}\quad \frac{\Gamma \longrightarrow \Delta_1, A, B, \Delta_2}{\Gamma \longrightarrow \Delta_1, B, A, \Delta_2}$$

contraction:

$$\textbf{left}\quad \frac{\Gamma_1, A, A, \Gamma_2 \longrightarrow \Delta}{\Gamma_1, A, \Gamma_2 \longrightarrow \Delta} \qquad \text{and} \qquad \textbf{right}\quad \frac{\Gamma \longrightarrow \Delta_1, A, A, \Delta_2}{\Gamma \longrightarrow \Delta_1, A, \Delta_2}$$

$\neg$: introduction:

$$\textbf{left}\quad \frac{\Gamma \longrightarrow \Delta, A}{\neg A, \Gamma \longrightarrow \Delta} \qquad \text{and} \qquad \textbf{right}\quad \frac{A, \Gamma \longrightarrow \Delta}{\Gamma \longrightarrow \Delta, \neg A}$$

$\wedge$: introduction:

$$\textbf{left}\quad \frac{A, B, \Gamma \longrightarrow \Delta}{A \wedge B, \Gamma \longrightarrow \Delta} \qquad \text{and} \qquad \textbf{right}\quad \frac{\Gamma \longrightarrow \Delta, A \quad \Gamma \longrightarrow \Delta, B}{\Gamma \longrightarrow \Delta, A \wedge B}$$

$\vee$: introduction:

$$\textbf{left}\quad \frac{A, \Gamma \longrightarrow \Delta \quad B, \Gamma \longrightarrow \Delta}{A \vee B, \Gamma, \longrightarrow \Delta} \qquad \text{and} \qquad \textbf{right}\quad \frac{\Gamma \longrightarrow \Delta, A, B}{\Gamma \longrightarrow \Delta, A \vee B}$$

cut:

$$\frac{\Gamma \longrightarrow \Delta, A \quad A, \Gamma \longrightarrow \Delta}{\Gamma \longrightarrow \Delta}$$

This system is p-equivalent to Frege systems, defined in [28]. When we add the additional rule that for a previously unused variable r and any formula ϕ, the sequents $r \longrightarrow \phi$ and $\phi \longrightarrow r$ may be introduced, and further stipulate that these extension atoms may not appear in the endsequent of a proof, we obtain a system equivalent to extended Frege systems, from the same paper.

Quantified propositional logic is what results when we add propositional quantifiers to our language. The semantics of $\forall x \phi(x, \overline{p})$ is that this formula is satisfied by a particular assignment if and only if $\phi(0, \overline{p}) \wedge \phi(1, \overline{p})$ is. Likewise the truth value of $\exists x \phi(x, \overline{p})$ is the same as that of $\phi(0, \overline{p}) \vee \phi(1, \overline{p})$. We can define a hierarchy of quantified Boolean semiformulas. The following is completely analogous to Definition 2.1.2:

Definition 2.2.3. *The classes Π_i^q and Σ_i^q are defined as follows:*

1. *$\Sigma_0^q = \Pi_0^q$ are the quantifier-free propositional semiformulas.*

2. *If ϕ is Σ_i^q or Π_i^q then it is also Σ_j^q and Π_j^q for all $j > i$.*

3. *If $\phi(x)$ is Σ_i^q then $\forall x\phi(x)$ is Π_{i+1}^q.*

4. *If $\phi(x)$ is Π_i^q then $\exists x\phi(x)$ is Σ_{i+1}^q.*

5. *If ϕ is Σ_i^q (Π_i^q) then $\neg\phi$ is Π_i^q (Σ_i^q respectively).*

6. *Σ_i^q and Π_i^q are closed under $\vee$ and $\wedge$.*

7. *Σ_i^q (Π_i^q) is closed under existential (universal) quantification.*

Now, the proof system G is obtained by augmenting the set of inference rules of LK with the following:

$\forall$: introduction:

$$\textbf{left} \quad \frac{A(B), \Gamma \longrightarrow \Delta}{\forall x A(x), \Gamma \longrightarrow \Delta} \qquad \text{and} \qquad \textbf{right} \quad \frac{\Gamma \longrightarrow \Delta, A(p)}{\Gamma \longrightarrow \Delta, \forall x A(x)}$$

$\exists$: introduction:

$$\textbf{left} \quad \frac{A(p), \Gamma \longrightarrow \Delta}{\exists x A(x), \Gamma \longrightarrow \Delta} \qquad \text{and} \qquad \textbf{right} \quad \frac{\Gamma \longrightarrow \Delta, A(B)}{\Gamma \longrightarrow \Delta, \exists x A(x)}$$

where B is any formula and the atom p replaced does not occur in the conclusion of the corresponding inference. G_i is G with the restriction that all formulas appearing in a proof must be Σ_i^q or Π_i^q. We shall consider G not only as a proof system for TAUT, but also for TAUT_i.

Boolean Programs

Boolean programs were introduced in [21] and are a way of specifying Boolean functions. Boolean programs are something like a generalization of the technique of using new atoms to replace part of a Boolean formula, which idea is the basis of extended Frege systems. As is the case with that system, and more so with the quantified propositional calculus, it appears that the use of Boolean programs allows formulas to be abbreviated significantly. The following definition is from that paper:

Definition 2.2.4 (Cook-Soltys)**.** *A* Boolean Program *P is specified by a finite sequence $\{f_1, ..., f_m\}$ of function symbols, where each symbol f_i has an associated arity k_i, and an associated defining equation*

$$f_i(\overline{p_i}) := A_i$$

where $\overline{p_i}$ is a list $p_1, ..., p_{k_i}$ of variables and A_i is a formula all of whose variables are among $\overline{p_i}$ and all of whose function symbols are among $f_1, ..., f_{i-1}$. In this context the definition of a formula *is:*

1. *0,1, and p are formulas, for any variable p.*
2. *If f is a k-ary function symbol in P and $B_1, ..., B_k$ are formulas, then $f(B_1, ..., B_k)$ is a formula.*
3. *If A and B are formulas, then $(A \wedge B), (A \vee B)$ and $\neg A$ are formulas.*

The semantics are as for propositional formulas, except that when evaluating an application $f_i(\overline{\phi})$ of a function symbol, the value is defined, using the defining equation, to be $A_i(\overline{\phi})$.

An interesting property of Boolean programs which demonstrates their comparability to quantified Boolean formulas is the following theorem from [21]:

Theorem 2.2.5 (Cook-Soltys)**.** *A Language L is in PSPACE iff L is computed by some uniform polynomial-size family of Boolean programs.*

BPLK

Definition 2.2.6 (BPLK)**.** *The system BPLK is like the propositional system LK, but with the following changes:*

1. *In addition to sequents, a proof also includes a Boolean program which defines functions. A BPLK-proof explicitly consists of a pair $\langle \pi, P \rangle$ of the proof (sequents) and the Boolean program defining the function symbols occurring in the sequents.*

2. *Formulas in sequents are formulas in the context of Boolean programs, as defined earlier.*

3. *If the Boolean program contains a definition of the form*

$$f(\overline{p}) := A(\overline{p}),$$

the new LK rules f : **left**

$$\frac{A(\overline{\phi}), \Gamma \longrightarrow \Delta}{f(\overline{\phi}), \Gamma \longrightarrow \Delta}$$

and f : **right**

$$\frac{\Gamma \longrightarrow \Delta, A(\overline{\phi})}{\Gamma \longrightarrow \Delta, f(\overline{\phi})}$$

may be used, where $\overline{\phi}$ are precisely as many formulas as $\overline{p}$ are variables.

4. *(***Substitution Rule***) The new inference rule* **subst**

$$\frac{\Delta(q, \overline{p}) \longrightarrow \Gamma(q, \overline{p})}{\Delta(\phi, \overline{p}) \longrightarrow \Gamma(\phi, \overline{p})}$$

may be used, where all occurrences of q have been substituted for.

The following is the main result of [53, 54]:

Theorem 2.2.7. *BPLK and G are polynomially equivalent.*

Finally, as this dissertation concerns propositional proof systems for large complexity classes, we would be remiss not to mention [38], in which the author describes how to transform a propositional proof system P into a system iP that is seemingly stronger in that it corresponds to an exponentially stronger theory. For example, iEF, constructed from extended Frege, EF, corresponds to the theory V_2^1 for exponential time. This construction, however, does not produce an alternate proof system for PSPACE. It is possible that by applying the construction to G_i, proof systems could be obtained for levels of the exponential-time hierarchy, but this is not clear. It does, however, by iterating the construction for EF, provide proof systems that would seem to correspond to the N-fold exponential-time theories of [12].

2.2.2 Complexity-Related Results

The primary motivation for studying propositional proof systems is the theorem of Cook and Reckhow [28] that NP=co-NP iff there exists a polynomially bounded proof system for propositional tautologies. There are, fortunately, many questions about these systems with less severe complexity-theoretic consequences than this one. One such question is how exactly the expressive power of a line of the proof relates to the relative efficiency of the system, which will be discussed in section 2.3.1. In this subsection we shall discuss how other modifications to a proof system, such as the restriction to treelike proofs or the addition of a substitution rule, affects its efficiency. We shall also talk about the witnessing problem for proofs of quantified tautologies.

Known Simulation Results

At the bottom of the G hierarchy of proof systems, which is already well above where the known lower bound results apply, we have G_0 which is polynomially equivalent to LK and Frege systems. It is also p-equivalent to its treelike subsystem G_0^* (since Frege and treelike Frege are p-equivalent [35]), something which is not known for G_i, $i > 0$. The next step up are Extended Frege and Substitution Frege systems, which are p-equivalent due to Dowd, and also Krajíček and Pudlák [39]. These are also both p-equivalent to G_1^* for proving quantifier-free tautologies. For $i > 0$, G_i p-simulates G_{i+1}^* for proofs of TAUT_i [36]. The converse simulation can also be shown, either directly or with the help of results such as those in section 2.3.1 and the conservativity of S_2^{i+1} over T_2^1. Another way of stating this last result is that substitution-G_i is p-equivalent to G_{i+1}^* for proofs of

TAUT_i for all i (including $i = 0$), and for $i > 0$, substitution is a derived rule in G_i [40].

Witnessing Problem for Quantified Propositional Proofs

The witnessing problem for quantified propositional proofs is the following: Given a proof of a quantified propositional tautology in Σ_i^q, and values for the free variables in the endsequent, find values for the outermost existentially quantified variables of the endsequent satisfying it. For G_1^* proofs, this problem is in P, and for G_1 proofs (of Σ_1^q tautologies), it is complete for PLS. It follows from [10] and the results in the next section that the witnessing problem for G_i is complete for an oracle version of PLS with a Σ_{i-1}^p oracle, defined in that paper, for each $i > 0$. For $i = 2$, the authors find an equivalent search problem they call GLS, for generalized local search. It is open to find more natural search problems for the rest of the cases, and it is also open to find any characterization of the witnessing problems for G_i proofs of Σ_j^q tautologies for $1 \leq j < i$. Another open problem is to find propositional proof systems whose witnessing problem corresponds to one of the other well-studied NP search classes. This can be done unnaturally by adding axioms asserting the totality of these search problems to EF.

2.3 Bounded Arithmetic and Propositional Proofs

In this section we discuss some connections between systems of bounded arithmetic and propositional proof systems.

2.3.1 Translations into Propositional Logic

The most important such connection is that some classes of theorems of some bounded arithmetic theories can be translated into families of propositional or quantified propositional tautologies. Depending on what the theory is and what class of formulas is translated, we can draw conclusions about the lengths of proofs of these families of tautologies in various propositional proof systems. Furthermore, by adding reflection principles, axioms stating the consistency of a propositional proof system, to a weaker theory, we can axiomatize a stronger theory corresponding to that proof system.

The first result of this form is due to Cook [25] who defines a translation from equations of PV to families of propositional formulas with polynomial-size EF proofs. Furthermore, any propositional proof system whose consistency PV can prove can be p-simulated by EF. Independently, Paris and Wilkie [46] gave a translation from bounded first-order formulas with a relation symbol R to families of propositional tautologies, and proved that if $I\Delta_0(R) \vdash \forall x\theta(x)$ then the translations of $\theta(x)$ have polynomial-size Frege proofs. Krajíček [37] extends this translation to handle second-order formulas and shows a similar relation between V_1^1 and polynomial-sized EF proofs, and between U_1^1 and quasipolynomial-sized Frege proofs.

Krajíček and Pudlák [40] extended Cook's result to show that whenever $A(a) \in \Sigma_i^b$ and $S_2^i \vdash A(a)$ (respectively, $T_2^i \vdash A(a)$), then the translations of $A(a)$ have polynomial-size G_i^* (respectively, G_i) proofs. Krajíček and Takeuti [42] showed a similar relation between U_2^1 and G, and such a result holds for BPLK as well [55].

2.3.2 Consistency Strength

Using the idea of Cook [25], [39], [42] and others define reflection principles $i - RFN(P)$ for each i and propositional proof system P, which states that P is sound for proofs of $\Sigma_i^q \bigcup \Pi_i^q$ tautologies. We have that for every i, $S_2^i \vdash i - RFN(G_i^*)$, $T_2^i \vdash i - RFN(G_i)$ and $U_2^1 \vdash i - RFN(G)$. Furthermore, for any proof system P such that one of the above theories, for example S_2^i, proves the reflection principle $j - RFN(P)$ for some j, the corresponding proof system, in this case G_i^*, p-simulates P for proofs of TAUT_j. In fact, every $\forall\Sigma_j^b$ consequence of S_2^i (T_2^i, U_2^1) follows from $S_2^1 + j - RFN(G_i^*)$ (G_i, G). For this reason, these reflection principles would be candidates for separating the theories.

Chapter 3

The Third-Order Viewpoint

In this chapter we introduce our third-order viewpoint, first for bounded arithmetic and then a calculus of functions.

3.1 Third-Order Bounded Arithmetic

We consider a three-sorted ("third-order") predicate calculus with free and bound variables of the first sort named $a, b, c, ...$ and $x, y, z, ...$, respectively, and free and bound variables of the second sort named $A, B, C, ...$ and $X, Y, Z, ...$, and likewise of the third sort named $\mathcal{A}, \mathcal{B}, \mathcal{C}, ...$ and $\mathcal{X}, \mathcal{Y}, \mathcal{Z}, ...$ The first sort are intended to represent natural numbers; the second, finite sets of natural numbers; and the third, finite sets of finite sets of natural numbers. We shall refer to the second sort as "strings" and the third sort as "superstrings", but formally they are both finite sets. In the standard model, variables of these types range over exactly these types of objects. A variable of unspecified sort will be denoted with a tilde, e.g.: $\tilde{x}$.

The language $\mathcal{L}_A^3 := \{0, 1, +, \cdot, |\cdot|, \in_2, \in_3, \leq, =_1, =_2\}$ of nonlogical symbols is the same as the set $\mathcal{L}_A^2$ for V^1 but with the addition of the third-order membership predicate $A \in_3 \mathcal{B}$. We write $\mathcal{A}(B)$ for $B \in_3 \mathcal{A}$ and $A(b)$ for $b \in_2 A$. We shall often omit the subscripts on '$=$' and '$\in$' as there is no danger of confusion. Note in particular the absence of the smash function symbol and third-order equality, as well as the length function being present for strings only: the expression $|X|$ is intended to represent 1 plus the largest element of the set X, or 0 if empty (the least upper bound of X). Such sets may be thought of interchangeably with ordinary finite binary strings under the following mapping, as in [16]: The set X represents the string with length $|X|-1$ whose i^{th} bit (for $0 \leq i < |X|-1$) is 1 exactly when $i \in_2 X$. This map is a bijection with the exception that the string corresponding to the empty set would be undefined, so we define it to be the empty string. Alternatively, our finite sets correspond to ordinary binary strings whose leading bit is 1, together with the empty string. We shall sometimes refer to $|X|$ as the **length** of X, although in fact X represents a string of length $|X|-1$.

Superstrings (or initial segments thereof) will also sometimes be thought of as strings of bits. The (ordinary) strings indexing the superstring are referred to as **bit-indices**. Since there is no length-function analogue for superstrings, the desired "length" (i.e., lexicographically maximal bit-index under consideration) will have to be specified separately.

Number terms are defined identically as in V^1, in particular not including any reference to third-order variables. Formulas are defined as usual, with the addition of the third-order variables and quantifiers on those variables. There is a hierarchy of classes $g\Sigma_i^{\mathcal{B}}$ and $g\Pi_i^{\mathcal{B}}$ of formulas in this language analogous to the hierarchies Σ_i^B and Π_i^B of second-order formulas: $g\Sigma_i^{\mathcal{B}}$ consists of those formulas with arbitrarily many bounded first- and second-order quantifiers, and at most i alternations of third-order quantifiers, the outer-most being **restricted**, i.e. equivalent to an existential quantifier. In contrast to this more general class, (**strict**) $\Sigma_i^{\mathcal{B}}$-formulas are those consisting of at most i alternating blocks of third-order quantifiers beginning with existential, followed by a formula with no third-order quantifiers; we shall also refer to a slightly more inclusive class of formulas called strict $\forall^2\Sigma_i^{\mathcal{B}}$, consisting of a single bounded universal second-order quantifier followed by a strict $\Sigma_i^{\mathcal{B}}$-formula. We shall refer to this class simply as $\forall^2\Sigma_i^{\mathcal{B}}$, omitting the explicit epithet “strict”.

Note that third-order quantifiers are not bounded, and in fact there is no apparent way to bound them (short of using an unbounded quantifier of a lower order) due to the lack of a length function. Fortunately, in the appropriate fragments of the theories we shall be concerned with, these variables will always be implicitly bounded, in the sense that the bounds on lower-order quantifiers will limit what part of the superstring actually affects the truth-value of a given formula.

3.2 Third-Order Computation and Function Calculus

In this section we introduce our framework of third-order computation. This includes a calculus of third-order functions that will be used later to obtain universal versions of some theories. The intent is to capture the nature of string-based computation defined by third-order theories of bounded arithmetic. For this reason, our primary focus is on classes of polynomially-bounded functions (from strings to strings) or similar, as this makes operations such as composition of functions more natural. We are consequently interested in our classes of functions somehow maintaining an exponential-size distinction between the three sorts, as do (standard) theories of bounded arithmetic. Furthermore, our intent when defining third-order complexity classes is that the third-order (superstring) arguments not count towards the resource limits of the machine.

Notwithstanding the above, we make the following definitions as general as we can. Functions in our setting will be strongly typed, meaning that each particular function has a fixed signature: some number of inputs of each of the three sorts (and possibly several inputs of each sort), and a single fixed output sort. The following definition specifies precisely the domains of the three sorts:

Definition 3.2.1. *First,* $\mathbb{D}^1 := \mathbb{N}$. *Second,* $\mathbb{D}^2 := \{S \subset \mathbb{N} : |S| < \infty\}$. *Finally,* $\mathbb{D}^3 = \{\mathcal{S} \subset \mathbb{D}^2 : |\mathcal{S}| < \infty\}$. *For convenience let* $\mathbb{D} = \mathbb{D}^1 \cup \mathbb{D}^2 \cup \mathbb{D}^3$.

(In this definition, $|\cdot|$ denotes set cardinality, and not the least upper bound function symbol). In other words, the three domains corresponding to the three sorts are: the natural numbers; finite sets of natural numbers; and finite sets of finite sets of natural numbers. Again we shall refer to these as numbers, strings, and superstrings. Note that these sorts are the same as the intended interpretations of the three sorts of variables in third-order bounded arithmetic. We shall use the same typographical conventions as for variables in third-order bounded arithmetic above to refer to members of these domains. Function symbols in our calculi will similarly be named from the lists $f, g, ...; F, G, ...;$ and $\mathcal{F}, \mathcal{G}, ...$ to indicate the sort of **the range of the function**. Functions or objects of unspecified sort will be named with a tilde such as $\tilde{a}$ or $\tilde{f}$. We shall also consider third-order predicates, which for simplicity we shall consider as 0-1 valued functions.

Definition 3.2.2. *Let* $\mathbb{E} = \mathbb{E}^1 \cup \mathbb{E}^2 \cup \mathbb{E}^3$ *be the set of all functions of fixed signature, categorized according to the sort of the output. The 0-1 valued functions (predicates) are referred to as* $\mathbb{E}^0 \subset \mathbb{E}^1$.

3.2.1 Computation of Functions

Now, expanding the usual definition of Turing machines computing functions from binary stings to binary strings, we define what it means for Turing machines to compute functions from $\mathbb{E}$. The broad overview and the intent to the definitions below is as follows: A machine computing a function must be able to receive the appropriate number of inputs of each of the three sorts, and also be able to output an object of the correct sort. String inputs (i.e., finite sets of natural numbers) are presented as ordinary binary strings and concatenated together. Numbers are presented in unary as strings. Finally, superstrings are presented on some kind of read-only, random-access input tape, analogously to how a

Turing machine accesses an oracle; recall that we intend for superstring inputs not to be included in the computation of resource limits. A machine then issues queries for desired bits of its superstring inputs by writing strings onto associated query tapes.

Numbers and strings are output as usual, but a superstring output is written to a special write-only output tape. This allows a space-bounded machine to output very long superstrings (longer than the space bound of the machine) simply by writing the bits in sequence; but observe that the run-time will be as long as the output, so a polynomial-time machine, for example, would be limited to polynomial-length superstring outputs. In some contexts, however, it is desirable and natural for a time-bounded machine to be able to "output" very long superstrings. One possible definition, and the one we adopt below, is for the machine to compute one requested bit of its superstring output; the results of many such queries over a range of query strings collectively form the superstring output of the machine. This will arise naturally in the context of polynomial-time hierarchy functions in third-order bounded arithmetic. The following definition formalizes the two kinds of computation from the discussion above:

Definition 3.2.3. *Let $(\mathbb{D}^1)^j \times (\mathbb{D}^2)^k \times (\mathbb{D}^3)^l$ be the domain of some function in $\tilde{f} \in \mathbb{E}$. Then a machine M* **computes** *the function if for every value of the parameters,*

$$M^{\mathcal{A}_1,\ldots\mathcal{A}_l}(1^{a_1}\#\ldots\#1^{a_j}\#A_1\#\ldots\#A_k)$$

outputs the value of the function onto a special write-only output tape, in unary in the case of a number-valued function. The notation $M^{\mathcal{A}}$ indicates that the superstring $\mathcal{A}$ is presented to M analogously to an oracle; namely, with read-only random access.

Not to be confused with the above, if the function is $\mathcal{F} \in \mathbb{E}^3$, we say that M **accepts** *$\mathcal{F}$ if $\mathcal{W} = \mathcal{F}(a_1, ..., a_j, A_1, ..., A_k, \mathcal{A}_1, ..., \mathcal{A}_l)$ is the value of the function on the indicated inputs and*

$$M^{\mathcal{A}_1,\ldots\mathcal{A}_l}(1^{a_1}\#\ldots\#1^{a_j}\#A_1\#\ldots\#A_k\#X)$$

accepts (or outputs 1) exactly when $X \in \mathcal{W}$.

3.2.2 Third-Order Complexity Classes

Below we give a general meta-definition that says how to convert an ordinary function or language class into a complexity class of third-order functions. We are interested primarily in **polynomially bounded** functions, which we now define in the context of third-order computation. The definition below says that the polynomial bound applies to a number output or the length of a string output, and is computed using only the number inputs and the lengths of the string inputs. Note that if there are only superstring inputs, then the definition implies that the bound is in fact a constant.

Definition 3.2.4. *A function $\tilde{f}$ with domain $(\mathbb{D}^1)^j \times (\mathbb{D}^2)^k \times (\mathbb{D}^3)^l$ is* **polynomially bounded** *if there is a fixed polynomial $p(n_1, ..., n_{j+k})$ such that for every $\overline{a} \in \mathbb{D}^1$, $\overline{A} \in \mathbb{D}^2$, $\overline{\mathcal{A}} \in \mathbb{D}^3$, either:*

1. $f(\overline{a}, \overline{A}, \overline{\mathcal{A}}) \leq p(\overline{a}, |A_1|, ..., |A_k|)$ *or*

2. $|F(\overline{a}, \overline{A}, \overline{\mathcal{A}})| \leq p(\overline{a}, |A_1|, ..., |A_k|)$,

as appropriate to the output sort of the function (number or string). Every function with superstring output sort is polynomially bounded.

Now we describe some specific cases of complexity classes we are interested in. The notation (various superscripts on the complexity classes) is from the meta-definition below.

First, FPSPACE$^+$ is the third-order analogue of PSPACE functions. It consists of those polynomially bounded functions computable by a machine in polynomial space (as a function of the string and (unary) number inputs only), where superstring outputs are written onto a write-only output tape, allowing exponential-length superstring outputs. The machine's queries to its superstring inputs must also be polynomially bounded (as a function of its inputs). FEXP$^+$ is similarly the polynomially bounded exponential-time functions with polynomially bounded access to superstring inputs. In contrast to $\mathrm{FPSPACE}^+$, the polynomial bound is an actual restriction as an exponential time machine could otherwise write exponentially large strings (either as output, or as a query to superstring inputs).

Now for the case of polynomial time, the class FP^+ defined analogously to $\mathrm{FPSPACE}^+$ and FEXP^+ has the property that superstring outputs have polynomial length, due to the time bound of the machines; however, the class P° of polynomially-bounded functions accepted by polynomial-time machines (see definition 3.2.3) does not have this restriction. For this reason, $\mathrm{FP}^+ \cup \mathrm{P}^\circ$ is in some contexts a more suitable third-order analogue of P. This is also the case for functions from levels $\square_i^p$ of the polynomial-time hierarchy, which are computed by polynomial-time machines with access to an oracle from Σ_i^p: The third-order class $(\square_i^p)^+$ is restricted to polynomially many bits in its superstring outputs and so $(\square_i^p)^+ \cup (\square_i^p)^\circ$ is a more appropriate definition.

As a final set of examples, the predicate classes $\mathrm{P}^\diamond$, $\mathrm{NP}^\diamond$, $(\Sigma_i^p)^\diamond$, $\mathrm{NEXP}^\diamond$ and $(\Sigma_i^{exp})^\diamond$ are 0-1 valued functions, and are the characteristic functions of machines from the corresponding ordinary complexity classes, modified with polynomially bounded access to superstring inputs.

We now state the meta-definition, which aims to collect and distill the notation and concepts from the preceding discussion. The definition is somewhat vague and intended only to give an approximate naming convention; specific instances of this definition will need clarification.

Definition 3.2.5 (Meta-Definition)**.** *Let FC be a complexity class of string functions with a well-understood semantics for oracle access such as a query tape, oracle gate, or similar. Then FC^+ denotes the class of polynomially bounded functions from $\mathbb{E}$* **computed** *by machines of the model of FC in the sense of definition 3.2.3, where queries to superstring inputs are polynomially bounded (and not counted in the space bound, if any, of the machine).*

Alternatively, let C be a complexity class of languages, again with an oracle semantics.

Then C° denotes the class of polynomially bounded functions $\mathcal{F} \in \mathbb{E}^3$ for which there is a machine $M_{\mathcal{F}}$ of the model of C **accepting** *$\mathcal{F}$ in the sense of definition 3.2.3, where again the queries to superstring inputs are polynomially bounded. In this case we ordinarily intend for the resource bounds of the computation to be determined by the arguments to the function only, and not the special query argument X to the superstring output.*

Finally, for C a class of languages, $C^{\diamond}$ denotes the class of 0-1 valued functions $f \in \mathbb{E}^0$ such that there is a machine M_f from the class of C accepts exactly when the value of f is 1, and issues only polynomially bounded queries to superstring inputs.

Some comments are in order concerning this definition. First, and most importantly, the third-order complexity classes discussed thus far, restricted to functions from strings to strings (or string predicates) are the usual complexity classes. There are nevertheless some interesting observations to be made: For example $\mathrm{P}^{\diamond} \neq \mathrm{NP}^{\diamond}$, as a predicate in the latter class can determine if a given superstring contains a 1 (up to a bound given by a string argument), while this predicate is clearly not in $\mathrm{P}^{\diamond}$. The usual argument for Savitch's theorem goes through, at least for (unrelativized) $\mathrm{NPSPACE}^{\diamond}$: configurations are still described by polynomial-sized strings, including queries to superstring inputs. We conclude that $\mathrm{PSPACE}^{\diamond} = \mathrm{NPSPACE}^{\diamond}$.

Now, in order to expand our discussion to the exponential-time hierarchy, we must first address relativizing classes of functions by adding oracles. It is most fervently desired that the reader not confuse these third-order oracles (our generalization of ordinary oracles) with the above use of oracle machines to receive third-order inputs. Our third-order version of oracle relativization is defined as follows:

Definition 3.2.6. *A* **third-order oracle** *Turing machine has a number of specified write-only query tapes, each one designated with a sort. The machine may write values on these tapes which are polynomially bounded, in the sense that the numbers (in unary), lengths of strings, and bit-indices of superstrings written are all bounded by fixed polynomials in the machine's (non-superstring) inputs. When the machine enters the* *special query state, these tapes are erased, and a value is returned to the machine by way of a special read-only reply tape (with random access in the case of a superstring-valued oracle).*

Now, the ordinary exponential-time hierarchy is defined by $\Sigma_i^{exp}(= \mathrm{NEXP}^{\Sigma_{i-1}^p})$ [33]. This is equal to Σ_i-TIME(exp), which are the languages computed by exponential-time alternating Turing machines with i alternations (starting with existential). Paralleling this definition, we can define the corresponding classes of 0-1 valued functions from $\mathbb{E}^0$. It is important to observe that the queries made of the Σ_{i-1}^p oracle by the NEXP machine in the standard definition are in general of **exponential** size. Our third-order oracle machines can also issue exponentially-long queries to their oracles, but these must be in the form of superstrings, as the string inputs to oracles are restricted to be polynomially bounded per our definition. Consequently the complexity class of the third-order oracle we use will be different.

We therefore define $(\Sigma_1^{exp})^\diamond$ =NEXP$^\diamond$ and $(\Sigma_i^{exp})^\diamond = (\mathrm{NEXP}^\diamond)^{(\Sigma_{i-1}^{exp})^\diamond}$. In other words, each higher level of the hierarchy is obtained by augmenting nondeterministic exponential time with a third-order oracle for the previous level. Since the queries to this oracle must be polynomially bounded (although this still allows exponential-length superstring inputs to the oracle), it can be seen that this relativization corresponds to unbounded access to an ordinary oracle from the appropriate level of the **quasi-polynomial-time** hierarchy (considered as a predicate on the superstring inputs): For example, if an NEXP machine writes string and superstring inputs of lengths $p(n)$ and $2^{p(n)}$ respectively to a third-order NEXP oracle, then the query can be answered in nondeterministic time $2^{(p(n))^k}$ for some k, which is exponential in $p(n)$. In terms of the length of the superstring input, $2^{p(n)}$, the quantity $2^{(p(n))^k}$ is quasi-polynomial.

In the hands of an NEXP machine, however, an unbounded (ordinary) oracle from some level of the quasi-polynomial-time hierarchy is no more powerful than one from the same level of the polynomial-time hierarchy, as the machine could simply make longer queries (i.e. $2^{(p(n))^k}$) of this latter oracle. Thus as predicates purely on strings, the levels of our hierarchy correspond precisely with the levels of the ordinary exponential-time hierarchy.

The function classes $(\square_i^{exp})^+ := (\mathrm{FEXP}^{(\Sigma_{i-1}^{exp})^\diamond})^+$ are the polynomially bounded functions computed by exponential-time Turing machines relativized with a third-order oracle for a function from $(\Sigma_i^{exp})^\diamond$, and similarly as functions purely of strings correspond to the usual $\square_i^{exp}$.

It should be noted that $(\Sigma_i^{exp})^\diamond = (\Pi_i^{exp})^\diamond$ seems to imply that the third-order exponential-time hierarchy collapses to the ith level, while this is not known for the ordinary case; The difference is that the assumption $\Sigma_i = \Pi_i$ in the third-order context is stronger, in that it covers also predicates on superstrings.

3.2.3 Recursion Theory of Functions

Now let us define some standard functions. The number functions $\{x + y, x \cdot y\}$, constants 0,1, etc. are as usual. The bit, string successor and concatenation functions $\{\mathrm{bit}(x, Y), s_0(X), s_1(X), X \frown Y\}$ are also standard, but they are operations on binary strings, while our string-like domain $\mathbb{D}^2$ consists of finite sets of natural numbers. We therefore define these functions to operate on the strings represented by the input finite sets, and to output the set representing the desired string. For example, the set $A = \{1\}$ represents the string 0 and has least upper bound $|A| = 2$. Therefore $s_0(A)$ is the set $\{2\}$ with least upper bound 3 that represents the string 00.

$\{|X|, X \in \mathcal{Y}, 1^x\}$ respectively give the least upper bound of the set (which is one more than the length of the string being represented by the set), the (0-1-valued) characteristic function of $\mathcal{Y}$, and a standard string of x bits (represented by a set of least upper bound $x + 1$). All of the functions described thus far are polynomially bounded.

We now define several operations on these functions. As our focus is on string functions as opposed to the standard recursion-theoretic viewpoint of number functions, we shall comment in each case on how these operations compare to standard operations on number functions.

First, the operation of **composition** defines a function $\tilde{f}(\bar{\tilde{x}}) = t$ by specifying a term t consisting of variables among $\bar{\tilde{x}}$ and other functions, constructed in such a way that arities and argument types are respected. The value of $\tilde{f}(\bar{\tilde{x}})$ is defined as the result of evaluating this term t in the obvious way (i.e., from the inside out). Observe that this operation allows permutation and renaming of variables.

Define $\tilde{f}$ (of any sort) by **limited recursion** from $\tilde{g}$, $\tilde{h}$ (also of any sort) and l by $\tilde{f}(0, ...) = \tilde{g}(...), \tilde{f}(x + 1, ...) = \tilde{h}(x, \tilde{f}(x, ...), ...)$ and either $\tilde{f}(x, ...) \leq l(x, ...)$ or $|\tilde{f}(x, ...)| \leq l(x, ...)$, as appropriate. This operation corresponds roughly to limited recursion on notation for number functions, as it iterates a function ($\tilde{h}$) a polynomial number of times subject to a bound on growth. **Recursion** is the same operation without the bound on growth.

Define $\tilde{f}$ by **limited doubling recursion** from $\tilde{g}$ and l by $\tilde{f}(0, \tilde{y}, ...) = \tilde{g}(\tilde{y}, ...), \tilde{f}(x+1, \tilde{y}, ...) = \tilde{f}(x, \tilde{f}(x, \tilde{y}, ...), ...)$ and either $\tilde{f}(x, \tilde{y}, ...) \leq l(x, ...)$ or $|\tilde{f}(x, \tilde{y}, ...)| \leq l(x, ...)$, as appropriate. This operation corresponds roughly to limited recursion for number functions, as it iterates a function ($\tilde{g}$) an exponential number of times (by doubling the number of nestings a polynomial number of times) subject to a bound on growth. **Doubling recursion** is the same operation without the bound on growth.

Define $\tilde{f}$ (of any sort) by **limited long recursion** from $\tilde{g}$, $\tilde{h}$ (also of any sort) and l by $\tilde{f}(1^0, ...) = \tilde{g}(...), \tilde{f}(X + 1, ...) = \tilde{h}(x, \tilde{f}(X, ...), ...)$ and either $\tilde{f}(X, ...) \leq l(X, ...)$ or $|\tilde{f}(X, ...)| \leq l(X, ...)$, as appropriate. This operation is similar to the previous one in that it iterates a function an exponential number of times; however, it differs in that the exponentially many iterations are performed directly by using a string as an exponential-length counter. This operation presupposes a suitable string successor function $X + 1$.

Define $\mathcal{F}$ by **limited 3-comprehension** from $g, h \in \mathbb{E}^1$ by $\mathcal{F}(..)(X) \leftrightarrow (|X| \leq g(..) \wedge h(X, ..) = 0)$.

For our purposes pertaining to theories of bounded arithmetic (in which superstrings are not bounded), it is important to distinguish these “pure” operations from bounded versions of them that only specify an initial segment of the superstring value of the function. The computational object so defined is now a multifunction, as there are many correct images of the multifunction for any set of parameters. For example, here is a bounded version of $\mathcal{F}$ defined by limited 3-comprehension from g and h: $|X| \leq g(...) \supset (\mathcal{F}(..)(X) \leftrightarrow h(X, ..) = 0)$.

It should be noted here that the recursion operations, as well as simple composition of functions, appear to be significantly more powerful when applied to superstring-valued functions. This is because in the composition of two such functions, the space may not be available to write down the intermediate value. A space-bounded computation model would then have to query the “inner” function many times (to retrieve bits of its output as needed) in order to compute the outer function. The composition of two polynomially bounded number- or string-valued functions can be computed using the sum of the time requirements (computing first one then the other function), while the required space does not increase. For superstring-valued functions, on the other hand, the time required for the composition as described seems in general to be the product of the time required for each component, while the space required is the sum. If space is not bounded then the intermediate results can be written in full, and thus time and space requirements are as for the composition of number- or string-valued functions.

At this point we can extrapolate a bit from Cobham [13] to state a characterization of polynomial-time functions:

Lemma 3.2.7. *The complexity class $FP^{+} \cup P^{\circ}$ is exactly the closure of the initial functions $I = \{0, 1, x + y, x \cdot y, 1^x, |X|, s_0(X), s_1(X), bit(x, Y), X \frown Y, X \in \mathcal{Y}\}$ under composition, limited 3-comprehension and limited recursion with the latter restricted to $\mathbb{E}^1 \cup \mathbb{E}^2$.*

Proof Sketch. Cobham's theorem states that the polynomial-time number functions are the closure of $I' = \{0, s_0(x), s_1(x), x\#'y\}$ under composition and limited recursion on dyadic notation. Here $x\#'y$ denotes the result of concatenating together $|y|$ copies of the dyadic notation for x, where $|y|$ is the length of the dyadic notation of y.

Now, the operations and functions above implicitly treat natural numbers as strings. In what follows, we will first show that the polynomial-time string functions are contained in the closure of I under the operations stated. We do so by defining explicitly string-based versions of the initial functions and operations of Cobham and then appealing to Cobham's theorem. Following that, we describe how to obtain all of $\mathrm{FP}^{+} \cup \mathrm{P}^{\circ}$. It is clear that I is contained in this class, which is closed under the indicated operations, and so the proof will be concluded.

The functions s_0 and s_1 in I are already string-based versions of Cobham's s_0 and s_1. With $\{0, 1, x+y, x \cdot y, 1^x, |X|\}$ we can construct numbers of polynomial magnitude (as a function of the lengths of string inputs) and thus also strings of polynomial length (although not exactly yet the string equivalent of $x\#'y$). A string selection function $\mathrm{sel}(x, Y, Z)$ can be defined using limited recursion up to $|Y|+|Z|$ to concatenate together the bits of either Y or Z depending on the value of x, thus selecting either Y or Z. Then using this function, limited recursion and bit, we can simulate the operation of limited recursion on dyadic notation. The string equivalent of $x\#'y$ is then easy to construct, at which point we have all the initial functions and operations of Cobham, and thus all the polynomial-time string functions.

We can easily extend our class of functions to number inputs by using 1^x to obtain a relevant string, and to number outputs by using $|X|$. To extend further to superstring inputs, we can directly simulate the operation of a polynomial-time Turing machine, at each step determining what bit of which superstring input is being queried, and using $X \in \mathcal{Y}$ to provide this bit to the next-state function of the machine. Finally, with limited 3-comprehension we can extend the class to superstring outputs.

$\square$

Observe that the operation of limited 3-comprehension uses a number-valued function as a predicate to obtain the bits of a superstring, and in this case produces exponential-length superstrings. To obtain the smaller class FP^+ (excluding P°), we would need an alternative way to obtain superstrings, as in this case 3-comprehension produces exponentially longer superstrings than functions from FP^+ can output. Alternatives include a function that produces a superstring from the bits of a given string, or a more sharply limited version of 3-comprehension. This problem is however, beyond the scope of this dissertation.

$\mathrm{FPSPACE}^+$ is contained in the closure of $\mathrm{FP}^+ \cup \mathrm{P}^\circ$ by limited recursion on $\mathbb{E}^3$, composition and limited 3-comprehension: First, a superstring-valued $\mathrm{FP}^+ \cup \mathrm{P}^\circ$ function can compute from the input of a PSPACE Turing machine the transition function of the machine as a table listing the next configuration for each given configuration. Another function in $\mathrm{FP}^+ \cup \mathrm{P}^\circ$ can compose such a function with itself by reading two (polynomial-sized) entries from this table. Therefore after applying limited recursion on these two functions we obtain a third that outputs the 2^x-step transition function and from this it is trivial to extract the value of the original PSPACE function. Since $\mathrm{FPSPACE}^+$ is closed under limited recursion (as each such operation increases the space requirements of a function by a polynomial factor), limited 3-comprehension and composition, we can conclude that this class is exactly the closure of the initial functions under these operations.

$\mathrm{FPSPACE}^+$ is alternatively characterized as the closure of $\mathrm{FP}^+ \cup \mathrm{P}^\circ$ under composition, limited 3-comprehension and limited doubling recursion restricted to $\mathbb{E}^1 \cup \mathbb{E}^2$. The step function of a PSPACE Turing machine can be iterated exponentially many times using these operations, and conversely $\mathrm{FPSPACE}^+$ is closed under limited doubling recursion as the recursion can be unwound with only a polynomial amount of additional space. This characterization is analogous to the one used in Dowd [29]: initial functions closed under limited recursion. Limited recursion in the context of number functions is of exponential length, as is limited doubling recursion in our setting. This in turn is reminiscent of $\mathcal{E}^2$, the second level of the Grzegorczyk hierarchy [30], which is defined similarly except with an initial function of linear growth rate as opposed to $x\#'y$; this was shown by Ritchie [52] to equal the linear space functions.

FEXP^+ is the closure of $\mathrm{FP}^+ \cup \mathrm{P}^\circ$ under composition, limited 3-comprehension and limited doubling recursion on $\mathbb{E}^3$: The step function of an exponential-time Turing machine can be iterated exponentially many times. See [11] for a previous recursion-theoretic characterization of the exponential-time number functions.

3.3 Representation Theorem

As a final section in this chapter, we connect third-order bounded arithmetic and computation with a representation theorem.

Theorem 3.3.1. *The predicates represented in the standard model by $\Sigma_0^{\mathcal{B}}$-formulas are precisely $PH^\diamond$; for $i \geq 1$ those represented by $g\Sigma_i^{\mathcal{B}}$- and $g\Pi_i^{\mathcal{B}}$-formulas (and also the strict versions of these classes) are precisely $(\Sigma_i^{exp})^\diamond$ and $(\Pi_i^{exp})^\diamond$, respectively.*

Proof. A $\Sigma_0^{\mathcal{B}}$-formula has some constant number of alternations of (bounded) string quantifiers. The problem of evaluating such a formula is therefore in the corresponding level of the (third-order) polynomial time hierarchy. Similarly, evaluating a $g\Sigma_i^{\mathcal{B}}$-formula is in third-order Σ_i-TIME(exp), which is $(\Sigma_i^{exp})^\diamond$: the formula is equivalent to one in strict quantifier syntax, and the i alternations of third-order quantifiers are evaluated by i alternations of nondeterministic exponential time, following which the remaining $\Sigma_0^{\mathcal{B}}$ subformula is evaluated deterministically in exponential time.

Conversely, standard techniques give, for an alternating Turing machine M (even one with superstring inputs), a $\Sigma_0^{\mathcal{B}}$-formula ϕ_M such that $\phi_M(\overline{\overline{x}}, \mathcal{C}_1, ..., \mathcal{C}_k)$ represents the predicate "$\mathcal{C}_1$,...,$\mathcal{C}_k$ code k exponential-length alternations of M on input $\overline{\overline{x}}$ and M accepts". Therefore $\exists \mathcal{C}_1 \forall \mathcal{C}_2 ... \phi_M(\overline{\overline{x}}, \overline{\mathcal{C}})$ represents the predicate "$M(\overline{\overline{x}})$ accepts" for Σ_i-TIME(exp)-machine M. Likewise, the characteristic predicate of a $\mathrm{PH}^\diamond$ machine is represented by a similar $\Sigma_0^{\mathcal{B}}$-formula with a constant number of additional leading bounded string quantifiers, which is altogether still $\Sigma_0^{\mathcal{B}}$. $\square$

Chapter 4

Third-order Theories

In this chapter we give the definition of several third-order theories and axiom schemes, and prove some basic properties of them.

4.1 Axiom Schemes and Theories

Our main theories are W_1^i and TW_1^i, intended to correspond to levels of the exponential-time hierarchy; they are parameterized by the type of induction. These theories are suggested by the RSUV isomorphism and are closely connected to U_2^i and V_2^i, respectively, although we do not claim an actual isomorphism (but one may hold with the unbounded domain versions of these theories).

For $i \geq 0$, W_1^i is a theory over $\mathcal{L}_A^3$. The axioms of W_1^i are B1-B14, L1, L2 and SE of [Cook/Kolokolova], (strict) $\forall^2\Sigma_i^{\mathcal{B}}$-IND and the following two comprehension schemes $\Sigma_0^{\mathcal{B}}$-2COMP:

$$(\exists Y \leq t(\overline{x}, \overline{X}))(\forall z \leq a)[\phi(\overline{x}, \overline{X}, \overline{\mathcal{X}}, z) \leftrightarrow Y(z)]$$

and $\Sigma_0^{\mathcal{B}}$-3COMP:

$$(\exists \mathcal{Y})(\forall Z \leq a)[\phi(\overline{x}, \overline{X}, \overline{\mathcal{X}}, Z) \leftrightarrow \mathcal{Y}(Z)],$$

where in each case $\phi \in \Sigma_0^{\mathcal{B}}$ subject to the restriction that neither Y nor $\mathcal{Y}$, as appropriate, occurs free in ϕ.

W_1^1 defined above is slightly different than the version published in CSL04 [55]; it includes a string equality symbol and extensionality axiom. This predicate is $\Delta_0^{\mathcal{B}}$-definable in the original version of the theory and can thus be conservatively added and used in all the axiom schemes. The unusual class of formulas for which we admit induction is in order for a replacement scheme to be provable; as a result of this scheme, W_1^i will ultimately admit full $g\Sigma_i^{\mathcal{B}}$-IND.

Define $\widehat{W_1^i}$ to be the analogous theory with the induction scheme restricted to (strict) $\Sigma_i^{\mathcal{B}}$ formulas. Note that $\widehat{W_1^0} = W_1^0$.

TW_1^i is defined identically as above, but with the following scheme named $\Sigma_i^{\mathcal{B}}$-SIND (string or set induction) in place of $\forall^2\Sigma_i^{\mathcal{B}}$-IND:

$$[\forall X, Y, Z((|Z| = 0 \supset \phi(Z)) \wedge (\phi(X) \wedge S(X,Y) \supset \phi(Y)))] \supset \forall Z \phi(Z)$$

for $\phi \in$(strict)$\Sigma_i^{\mathcal{B}}$, where $S(X,Y)$ is the following formula expressing that Y is the lexicographically next finite set after X:

$$\begin{aligned}|Y| \leq |X| + 1 \wedge \exists i \leq |Y|[Y(i) \wedge \neg X(i) \wedge \forall j < i(X(j) \wedge \neg Y(j)) \wedge \\ \forall j \leq |Y|(i < j \supset (X(j) \leftrightarrow Y(j)))].\end{aligned}$$

(This formulation is due to Phuong Nguyen). It will also turn out that TW_1^i admits (string) induction on the more general class of formulas due to a replacement scheme.

TTW_1^i is yet another theory in this vein, with a yet stronger induction scheme named $\Sigma_i^{\mathcal{B}}$-SSIND ("superstring" induction). Note that since (by design) there is no way to bound a third-order object, the scheme refers to a term t, and restricts its attention to the first 2^t bits of the objects. It is intended that this t be some crucial bound from ϕ. The scheme is:

$$[\forall \mathcal{X}, \mathcal{Y}, \mathcal{Z}((\forall X \leq t \neg \mathcal{Z}(X)) \supset \phi(\mathcal{Z})) \wedge (\phi(\mathcal{X}) \wedge S_3(\mathcal{X}, \mathcal{Y}, t) \supset \phi(\mathcal{Y}))] \supset \forall \mathcal{Z} \phi(\mathcal{Z})$$

for $\phi \in$(strict)$\Sigma_i^{\mathcal{B}}$, where $S_3(\mathcal{X}, \mathcal{Y}, z)$ is the formula

$$\exists Z \leq z[\mathcal{Y}(Z) \wedge \neg\mathcal{X}(Z) \wedge$$
$$\forall X \leq z(L_2(X,Z) \supset \mathcal{X}(X) \wedge \neg\mathcal{Y}(X)) \wedge \forall X \leq z(L_2(Z,X) \supset \mathcal{X}(X) \leftrightarrow \mathcal{Y}(X)))$$

and $L_2(X,Y)$ is the formula

$$\exists i \leq |X|[Y(i) \wedge \neg X(i) \wedge \forall j \leq |Y|(i < j \supset (X(j) \leftrightarrow Y(j)))],$$

which is intended to mean that $X < Y$, considered as large numbers.

The scheme $\Sigma_0^{\mathcal{B}}$-superstring-recursion is the following:

$$\exists \mathcal{X} \phi^{\mathrm{rec}}(S, \mathcal{X}),$$

where $\phi(Y, \mathcal{X}) \in \Sigma_0^{\mathcal{B}}$, and

$$\phi^{\mathrm{rec}}(x, \mathcal{X}) \equiv \forall Y \leq |S|(L_2(Y,S) \supset (\mathcal{X}(Y) \leftrightarrow \phi(Y, \mathcal{X}^{<Y}))),$$

where $\mathcal{X}^{<Y}$ is a chop function (i.e., $\mathcal{X}^{<Y}(Z)$ abbreviates the subformula $L_2(Z,Y) \wedge \mathcal{X}(Z)$). ϕ (and therefore also ϕ^{rec}) may have other free variables than the displayed ones, but ϕ must have distinguished string and superstring free variables Y and $\mathcal{X}$. ϕ^{rec} then has $\mathcal{X}$ free as well as a new variable S. This scheme is analogous to that from [6] and follows the presentation from [27].

The scheme $\Sigma_0^{\mathcal{B}}$-superstring-halfrecursion is the following:

$$\exists \mathcal{X} \phi^{\mathrm{hrc}}(S, \mathcal{X}),$$

where $\phi(Y, \mathcal{X}) \in \Sigma_0^{\mathcal{B}}$, and

$$\phi^{\mathrm{hrc}}(S, \mathcal{X}) \equiv \forall Y \leq |S|(L_2(Y,S) \supset (\mathcal{X}(Y) \leftrightarrow \phi(Y, \mathcal{X}^{<Y/2}))),$$

where $\mathcal{X}^{<Y/2}$ is a chop function returning the first $\frac{Y}{2}$ (as a number) bits of $\mathcal{X}$. ϕ and ϕ^{rec} have the same free-variable conventions and requirements as in the superstring recursion scheme. Then HW_1^0 is the theory W_1^0 with the addition of the $\Sigma_0^{\mathcal{B}}$-superstring-halfrecursion scheme.

4.2 Third-Order Parikh's Theorems

In this section we prove a generalization of Parikh's theorem for third-order theories. First, some definitions:

Definition 4.2.1. *A formula is* **2-bounded** *if all of its first- and second-order quantifiers are bounded. (It may contain arbitrary third-order quantifiers).*

Let T be a theory extending W_1^0 and $\mathcal{L} \supseteq \mathcal{L}_A^3$ be the vocabulary of T. Then T is a **2-bounded theory** *if it is axiomatized by 2-bounded formulas.*

Definition 4.2.2. *Let $M = \langle M_1, M_2, M_3 \rangle$ be a model of B1-B14, L1, L2, SE. Analogously to the first-order case, a 2-cut in M is any subset $I = \langle I_1 \subseteq M_1, I_2 \subseteq M_2, I_3 = M_3 \rangle$ closed under $x + 1$ and $\leq$ (for numbers and strings). This last point means that if $b \in I_1$ and $M \models a \leq b$ for $a \in M_1$ (or $M \models |A| \leq b$ for $A \in M_2$), then $a \in I_1$ (respectively, $A \in I_2$). For a string $A \in M_2$, it is equivalent to say that if $|A| \in I_1$ then $A \in I_2$.*

This is denoted $I \subseteq_e^2 M$.

Lemma 4.2.3. *Let M be a third-order structure with vocabulary $\mathcal{L}$ and and $I \subseteq_e^2 M$ be a 2-cut of M closed under all the function symbols in $\mathcal{L}$. Finally, let $\phi(\overline{a}, \overline{A}, \overline{\mathcal{A}})$ be a 2-bounded formula with all free variables displayed, and $\overline{b}, \overline{B}, \overline{\mathcal{B}} \in I$. Then*

$$I \models \phi(\overline{b}, \overline{B}, \overline{\mathcal{B}}) \qquad \textit{iff} \qquad M \models \phi(\overline{b}, \overline{B}, \overline{\mathcal{B}}).$$

Proof Sketch. This lemma is proved by induction on the quantifier complexity of ϕ. The base case is quantifier-free formulas, and is clear, as all parameters are in the cut.

For the induction step, consider $I \models \forall X \leq t\phi(X)$. All parameters (including those in t), not shown, are from I. Then $I \models \phi(B)$ for each $B \leq t$. But then $M \models \phi(B)$ for each $B \leq t$ in M as all such elements are already in I, and so $M \models \forall X \leq t\phi(X)$. The other direction ($M \models \forall X \leq t\phi(X) \implies I \models \forall X \leq t\phi(X)$) is easier as the range of the universal quantifier is decreased.

The case of a first-order quantifier is very similar, and existential quantifiers are handled symmetrically. The case of a third-order quantifier is straightforward, as the range of the quantifier remains the same in M or I. $\square$

Note that the above lemma does not require any assumption about function symbols being bounded by monotone terms. It also does not restrict the sorts of the range or any component of the domain of a function symbol.

At this point it is apparent that the open axioms B1-B13, L1 and L2, the predecessor axiom B14, SE and the comprehension and recursion schemes are satisfied in any $\mathcal{L}$-closed 2-cut of any model of any 2-bounded theory T, as all are 2-bounded. In fact, the same is true of the induction schemes, even the sharply bounded ones. This is because they all have 2-bounded versions. The 2-bounded B-$\Sigma_i^{\mathcal{B}}$-SIND is:

$$[\forall Z \leq 0 \forall X \leq |W| \forall Y \leq |W| (\phi(Z) \wedge (\phi(X) \wedge S(X,Y) \supset \phi(Y)))] \supset \phi(W)$$

and the 2-bounded B-$\Sigma_i^{\mathcal{B}}$-IND is:

$$[\phi(0) \wedge \forall x \leq w(\phi(x) \supset \phi(x+1))] \supset \phi(w).$$

These bounded induction schemes logically imply the unbounded versions. Conversely, the unbounded schemes prove the bounded ones: for example, $\Sigma_i^{\mathcal{B}}$-SIND on the formula $|X| \leq |W| \supset \phi(X)$ gives

$$[\forall X, Y, Z((|Z| = 0 \supset (|Z| \leq |W| \supset \phi(Z))) \wedge ((|X| \leq |W| \supset \phi(X)) \wedge S(X,Y) \supset (|Y| \leq |W| \supset \phi(Y))))] \supset \forall Z(|Z| \leq |W| \supset \phi(Z)).$$

By strengthening the hypothesis,

$$[\forall Z \leq 0 \forall X, Y(\phi(Z) \wedge ((|X| \leq |W| \supset \phi(X)) \wedge S(X,Y) \supset (|Y| \leq |W| \supset \phi(Y))))] \supset \forall Z(|Z| \leq |W| \supset \phi(Z))$$

and again by the fact that if $S(X,Y)$ then $|X| \leq |Y|$,

$$[\forall Z \leq 0 \forall X \leq |W| \forall Y \leq |W| (\phi(Z) \wedge (\phi(X) \wedge S(X,Y) \supset \phi(Y)))] \supset \forall Z (|Z| \leq |W| \supset \phi(Z)).$$

Finally,

$$[\forall Z \leq 0 \forall X \leq |W| \forall Y \leq |W| (\phi(Z) \wedge (\phi(X) \wedge S(X,Y) \supset \phi(Y)))] \supset \phi(W),$$

which is B-ϕ-SIND.

Thus we have proven that all the theories described above in section 4 are in fact 2-bounded.

A corollary of the previous lemma:

Corollary 4.2.4. *Let T be any 2-bounded extension of W_1^0 and let M be a model of T. Let $I \subseteq_e^2 M$ be a 2-cut of M closed under all the function symbols in $\mathcal{L}$. Then the (2-bounded) axioms of T are satisfied by I, and consequently, $I \models T$.*

We require one additional definition before stating Parikh's theorem for first- and second-order existential quantifiers:

Definition 4.2.5. *Let T be a three-sorted theory with vocabulary $\mathcal{L} \supseteq \mathcal{L}_A^3$ containing the open axioms of W_1^0. We say that T has* **monotone 2-bounding** *if the following hold:*

1. *For every number-valued function symbol $f \in \mathcal{L}$, there is a number term t_f of $\mathcal{L}$ such that*

   $$T \vdash \overline{\tilde{a}} \leq \overline{b} \supset f(\overline{\tilde{a}}) \leq t_f(\overline{b}),$$

   where $\overline{\tilde{a}}$ is a list of variables of any order and $\overline{b}$ is a list of number variables, and $\overline{\tilde{a}} \leq \overline{b}$ abbreviates a conjunction of subformulas of the form $a_i \leq b_i$ or $|A_i| \leq b_i$, as appropriate, for each $\tilde{a}_i$ not a third-order variable.

2. *For every string-valued function symbol $F \in \mathcal{L}$, there is a term t_F of $\mathcal{L}$ such that $T \vdash \overline{\tilde{a}} \leq \overline{b} \supset |F(\overline{\tilde{a}})| \leq t_F(\overline{b})$.*

Theorem 4.2.6 (Third-Order Parikh's Theorem). *Let $\phi(\overline{\tilde{x}})$ be a 2-bounded formula, all free variables displayed, and T a 2-bounded extension of W_1^0 with vocabulary $\mathcal{L}$ and monotone 2-bounding.*

Further, assume that $T \vdash \forall\overline{\tilde{x}}\exists\tilde{y}\phi(\overline{\tilde{x}},\tilde{y})$, where $\overline{\tilde{x}}$ are of any sort and $\tilde{y}$ is first- or second-order. Then there is some term t such that $T \vdash \forall\overline{\tilde{x}}\exists\tilde{y} \leq t(\overline{\tilde{x}})\phi(\overline{\tilde{x}},\tilde{y})$.

Proof. This theorem is proved by a compactness argument, and we refer the reader to the standard model-theoretic arguments in [36] and [31] which we emulate here. Assume the hypothesis of the theorem, and furthermore that $T \not\vdash \forall\overline{\tilde{x}}\exists\tilde{y} \leq t(\overline{\tilde{x}})\phi(\overline{\tilde{x}},\tilde{y})$ for any number term t. Then by compactness the theory

$$T' = T + \{\forall\tilde{y} \leq t(\overline{\tilde{c}})\neg\phi(\overline{\tilde{c}},\tilde{y}) : t \text{ any term of } \mathcal{L}\}$$

is consistent with $\overline{\tilde{c}}$ new constants of the appropriate sorts.

Now, let $M \models T'$ and define $I \subseteq_e^2 M$ by $b \in I_1$ (respectively, $B \in I_2$) iff there is a term t such that $M \models b < t(\overline{\tilde{c}})$ ($M \models |B| < t(\overline{\tilde{c}})$). (And $I_3 = M_3$). It is evident that I is indeed a 2-cut of M, but for I to be $\mathcal{L}$-closed, it is essential at this point for T to have monotone 2-bounding. Otherwise, there could be some $b \in I$ by virtue of $M \models b \leq t(\overline{\tilde{c}})$, yet $f(b) \notin I$ for f is not monotone. However, our assumption ensures that $M \models f(b) \leq t_f(t(\overline{\tilde{c}}))$. The monotone 2-bounding assumption implies that applying function symbols to elements in I (which are bounded by terms in $\overline{\tilde{c}}$) produces elements which are also bounded, and so already in I.

I is then a model for T by the corollary, and yet $I \models \exists\overline{\tilde{x}}\forall\tilde{y}\neg\phi(\overline{\tilde{x}},\tilde{y})$, a contradiction. ∎

4.3 Generalized Definability

In this section we present some important definitions concerning definability of functions in third-order theories. These definitions generalize the standard kind of definability in order to address third-order computation.

Definition 4.3.1. *Let T be a theory with vocabulary $\mathcal{L} \supseteq \mathcal{L}_A^3$ and Φ a set of $\mathcal{L}$-formulas. Then a function $\tilde{f} \in \mathbb{E}^1 \cup \mathbb{E}^2$ is* **Φ-definable** *in T if there is some $\phi \in \Phi$ such that:*

1. $T \vdash \forall \overline{x}, \overline{X}, \overline{\mathcal{X}} \exists! \tilde{y} \phi(\overline{x}, \overline{X}, \overline{\mathcal{X}}, \tilde{y})$

2. $\phi(\overline{x}, \overline{X}, \overline{\mathcal{X}}, \tilde{f}(\overline{x}, \overline{X}, \overline{\mathcal{X}}))$ *is true in the standard model for all values of the parameters.*

The defining axiom for $\tilde{f}$ is then

$$\tilde{f}(\overline{x}, \overline{X}, \overline{\mathcal{X}}) = \tilde{y} \leftrightarrow \phi(\overline{x}, \overline{X}, \overline{\mathcal{X}}, \tilde{y})$$

Now, for superstring-valued functions we must use a slightly weaker kind of definability. This is because there is no way to bound a superstring, and thus no (bounded) way to assert the equality of two superstrings. Furthermore, our comprehension axioms assert the existence of certain superstrings but specify only an initial segment of their bits. Thus the following definitions:

Definition 4.3.2. *For superstring variables $\mathcal{X}$ and $\mathcal{Y}$ and term t, let $\mathcal{X} =_t \mathcal{Y}$ abbreviate the $\Sigma_0^{\mathcal{B}}$-formula $\forall Z \leq t(\mathcal{X}(Z) \leftrightarrow \mathcal{Y}(Z))$.*

Definition 4.3.3. *Let T, $\mathcal{L}$ and Φ be as above. Let $t(\overline{x}, \overline{X})$ be a number term over $\mathcal{L}$, which in the standard model bounds the* **lengths of bit-indices** *in the output of a function $\mathcal{F}(\overline{x}, \overline{X}, \overline{\mathcal{X}}) \in \mathbb{E}^3$. Then $\mathcal{F}$ is* **length 2^t Φ-definable in** *T if there is some $\phi \in \Phi$ such that:*

1. $T \vdash \forall \overline{x}, \overline{X}, \overline{\mathcal{X}} \exists \mathcal{Y} \phi(\overline{x}, \overline{X}, \overline{\mathcal{X}}, \mathcal{Y})$

2. $T \vdash \forall \overline{x}, \overline{X}, \overline{\mathcal{X}}, \mathcal{Y}, \mathcal{Y}'[\phi(\overline{x}, \overline{X}, \overline{\mathcal{X}}, \mathcal{Y}) \wedge \phi(\overline{x}, \overline{X}, \overline{\mathcal{X}}, \mathcal{Y}') \longrightarrow \mathcal{Y} =_t \mathcal{Y}']$

3. $T \vdash \forall \overline{x}, \overline{X}, \overline{\mathcal{X}}, \mathcal{Y}, \mathcal{Y}'[\mathcal{Y} =_t \mathcal{Y}' \wedge \phi(\overline{x}, \overline{X}, \overline{\mathcal{X}}, \mathcal{Y}) \longrightarrow \phi(\overline{x}, \overline{X}, \overline{\mathcal{X}}, \mathcal{Y}')]$

4. $\phi(\overline{x}, \overline{X}, \overline{\mathcal{X}}, \mathcal{F}(\overline{x}, \overline{X}, \overline{\mathcal{X}}))$ *is true in the standard model for all values of the parameters.*

The defining axiom for $\mathcal{F}$ is then

$$(\mathcal{Y} =_t \mathcal{F}(\overline{x}, \overline{X}, \overline{\mathcal{X}})) \leftrightarrow \phi(\overline{x}, \overline{X}, \overline{\mathcal{X}}, \mathcal{Y})$$

If $\mathcal{F}$ is length 2^t Φ-definable in T for some true bound $t(\overline{x}, \overline{X})$, then we say $\mathcal{F}$ is Φ-definable in T.

In sum, a length 2^t-definable function in a theory T is provably total, provably unique up to 2^t bits, and the formula defining the graph of the function is provably insensitive to bits beyond this bound. Furthermore, the true value of the function satisfies the graph. The defining axiom specifies the bits of the value of the function up to length 2^t, but beyond this point it is undefined (although also not relevant, as far as the graph is concerned).

Note that for suitable theories, a length 2^t-definable function $\mathcal{F}$ is also length 2^s-definable for any s that is provably larger than t, either by modifying the defining formula to check that the extra bits are zeroes, or by composing the function with an extender function that adds the extra zeroes. This will be discussed further in section 6.

Note further that a superstring-valued function definable in a theory in the sense of the previous definition can be conservatively added to the theory in the same way as the more concretely definable functions of the previous definition (although we do not claim at this point that the theory admits its axiom schemes such as induction or comprehension in the augmented language). The defining axiom is deliberately vague about specifying the value of the function, and therefore any (say) string-valued function G defined using (i.e., as a function of) $\mathcal{F}$ must provably depend only on the bits of the output of $\mathcal{F}$ actually fixed by the defining axiom – otherwise the uniqueness clause of the definition of G will presumably not hold.

It may be desirable in some cases to assert that a definable function with a superstring argument is insensitive in this way to variations in its superstring argument. The following definition formalizes this concept:

Definition 4.3.4. *Let* $\tilde{f}(\mathcal{X}, \overline{y}, \overline{Y}, \overline{\mathcal{Y}}) \in \mathbb{E}^1 \cup \mathbb{E}^2$ *be a definable function of a theory* T *(as above) and* $t(\overline{y}, \overline{Y})$ *a term of* $\mathcal{L}$*, the language of* T*. Then* $\tilde{f}$ *is* **insensitive to** $\mathcal{X}$ **beyond** t *if*

$$T \vdash \mathcal{X} =_t \mathcal{X}' \longrightarrow \tilde{f}(\mathcal{X}, \overline{y}, \overline{Y}, \overline{\mathcal{Y}}) = \tilde{f}(\mathcal{X}', \overline{y}, \overline{Y}, \overline{\mathcal{Y}})$$

Similarly, a function $\mathcal{F}(\mathcal{X}) \in \mathbb{E}^3$ *that is length* 2^s *definable in* T *is* **insensitive to** $\mathcal{X}$ **beyond** t *if*

$$T \vdash \mathcal{X} =_t \mathcal{X}' \longrightarrow \mathcal{F}(\mathcal{X}) =_s \mathcal{F}(\mathcal{X}')$$

If $\tilde{f}(\overline{y}, \overline{Y}, \mathcal{Y}_1, ..., \mathcal{Y}_k)$ *has* k *superstring arguments and is* $t_i(\overline{y}, \overline{Y})$*-insensitive to* $\mathcal{Y}_i$ *for each* I*, then we say that* $\tilde{f}$ *is* $(t_1, ..., t_k)$*-insensitive.*

A restricted version of an important standard property of bounded arithmetic holds for third-order theories:

Lemma 4.3.5. *Let* T *be a theory with vocabulary* $\mathcal{L} \supseteq \mathcal{L}_A^3$ *and* $\tilde{f} \in \mathbb{E}^1 \cup \mathbb{E}^2$ *be a* $\Sigma_0^{\mathcal{B}}$*-definable function in* T*. Then* T^+ *(over* $\mathcal{L}^+ = \mathcal{L} \cup \{\tilde{f}\}$*), obtained by adding the defining axioms for* $\tilde{f}$ *to* T*, is a conservative extension of* T*. Furthermore, if* A *is a* $\Sigma_i^{\mathcal{B}}$*- or* $\Pi_i^{\mathcal{B}}$*-formula over* $\mathcal{L}^+$*, then there is a formula* A^- *of the same class but over* $\mathcal{L}$ *such that*

$$T \vdash (A \leftrightarrow A^-)$$

The lemma is proved as for Theorem 2.2 of Buss [3], although much simpler as the latter theorem applies also to Σ_1^b-definable functions. Unfortunately, the proof does not seem to go through in the case of a superstring-valued function from $\mathbb{E}^3$: since the defining axiom does not specify a unique value for the output of the function, the translated formula incorporating the defining axiom is not obviously equivalent to the original one. However, is is possible that an additional assumption on the original formula concerning its sensitivity to the value of the function (i.e., less than the defined length) would suffice. This could take the form of Lemma 10.9 of [3].

An analogue to the full Theorem 2.2 of Buss would be that $\Sigma_1^{\mathcal{B}}$-definable functions could be conservatively added; however, this only seems to be true of an appropriate form of replacement (see section 4.5) is available, as it is in Buss's theories.

4.4 Preliminary Facts About the Theories

In this section we enumerate some simple results concerning the various theories described above. Most of these will be explained further in the relevant parts of future chapters.

Lemma 4.4.1. *W_1^0 is a conservative extension of V, the two-sorted theory for the poly-time hierarchy.*

Proof outline. The only axioms stating the existence of third-order elements are the comprehension axioms. Any model of V can therefore be expanded to a model of W_1^0 merely by adding third-order elements to satisfy each comprehension instance ($\Sigma_0^{\mathcal{B}}$ formula with parameters from the model under construction). For a general $\Sigma_0^{\mathcal{B}}$-3COMP instance with free variables, we must supply an object satisfying the instance for each set of parameters from the model assigned to the free variables. $\Sigma_0^{\mathcal{B}}$ formulas are closed under substitution of formulas for free third-order variables, so ultimately each comprehension instance unwinds to a $\Sigma_0^{\mathcal{B}}$ formula. In the end, adding these third-order elements will not affect the truth-value of purely second-order formulas. □

The $\Sigma_0^{\mathcal{B}}$-definable functions of W_1^0 from $\mathbb{E}^1 \cup \mathbb{E}^2$, of number and string arguments, are thus the usual polynomial-time hierarchy functions. The $\Sigma_1^{\mathcal{B}}$-definable functions of W_1^0 are contained in FPH$^+ \cup$FPH$^\circ$ (the third-order generalization), as the usual witnessing argument for V can be extended to handle third-order existential quantifiers, which arise only because of the comprehension axioms and introduction rules, and in either case are witnessed by FPH$^+ \cup$FPH$^\circ$ functions.

The $\Sigma_1^{\mathcal{B}}$-definable functions of W_1^1 are exactly FPSPACE$^+$ [55]. As it turns out, The $\Sigma_1^{\mathcal{B}}$-definable functions of $\widehat{W_1^1}$ (i.e., with more restricted induction) are also exactly

FPSPACE$^+$; see chapter 5. A big question is whether or not this theory proves the replacement schemes from the next section.

The $\Sigma_1^{\mathcal{B}}$-definable functions of TW_1^0 are also only FPH$^+\cup$FPH$^\circ$. TW_1^0 is a conservative extension of TV, which is V with the addition of Σ_∞^B-SIND, but $TV = V$. In fact,

Lemma 4.4.2. $W_1^0 = TW_1^0$

Proof. We use the same "shortening of cuts" technique used to prove $S_2^{i+1} \supseteq T_2^i$ in order to show that $W_1^0 \vdash \Sigma_0^{\mathcal{B}}$-SIND:

Let $\phi(X) \in \Sigma_0^{\mathcal{B}}$ and A a parameter. Define

$$\psi(x) := \forall X \leq x \forall Y \leq |A| \forall Z \leq |A| (\phi(Y) \wedge \text{Plus}(X, Y, Z) \longrightarrow \phi(Z)).$$

(For a suitable function symbol Plus). Now, trivially $W_0^1 \vdash \psi(0)$. Also,

$$W_1^0 \vdash (\forall X \forall Y (\phi(X) \wedge S(X, Y) \longrightarrow \phi(Y))) \longrightarrow (\psi(x) \longrightarrow \psi(x+1))$$

by considering the two cases of the low-order bit of X (in $\psi(x+1)$) and applying the induction step of $\Sigma_0^{\mathcal{B}}$-SIND if necessary. Thus by $\Sigma_0^{\mathcal{B}}$-IND, $W_1^0 \vdash \psi(|A|)$. This and the remaining hypothesis of $\Sigma_0^{\mathcal{B}}$-SIND, $\forall X(|X| = 0 \longrightarrow \phi(X))$, imply $\phi(A)$. Therefore $W_1^0 \vdash \Sigma_0^{\mathcal{B}}$-SIND and thus $W_1^0 = TW_1^0$. $\square$

The $\Sigma_1^{\mathcal{B}}$-definable functions of TTW_1^0 are EXP. This is because TTW_1^0 proves $\Sigma_0^{\mathcal{B}}$-superstring-recursion:

$$\exists \mathcal{X} \phi^{\text{rec}}(Y, \mathcal{X}),$$

where $\phi \in \Sigma_0^{\mathcal{B}}$, and

$$\phi^{\text{rec}}(S, \mathcal{X}) \equiv \forall Y \leq |S| (\mathcal{X}(Y) \leftrightarrow \phi(Y, \mathcal{X}^{<Y})),$$

where $\mathcal{X}^{<Y}$ is a chop function. See chapter 5 for details.

Finally, concerning HW_1^0:

Lemma 4.4.3. $W_1^1 \vdash HW_1^0$

Proof. Recall that $\Sigma_0^{\mathcal{B}}$-superstring-halfrecursion is

$$\exists \mathcal{X} \forall Y \leq x(\mathcal{X}(Y) \leftrightarrow \phi(Y, \mathcal{X}^{<Y/2})).$$

for $\phi \in \Sigma_0^{\mathcal{B}}$. W_1^1 can prove this directly by induction on x. The induction step, from x to $x+1$, involves a single application of $\Sigma_0^{\mathcal{B}}$-3COMP to produce a new superstring consisting of the current one plus a new segment:

$$\exists \mathcal{X}' \forall Y \leq x+1(\mathcal{X}'(Y) \leftrightarrow (|Y| \leq x \wedge \mathcal{X}(Y)) \vee (|Y| = x+1 \wedge \phi(Y, \mathcal{X}^{<Y})))$$

W_1^1 straightforwardly proves that this new superstring satisfies the halfrecursion up to $x+1$. □

Furthermore, the $\Sigma_1^{\mathcal{B}}$-definable functions of HW_1^0 are exactly FPSPACE$^+$; definability and witnessing theorems are found in the following chapters.

4.5 $\Sigma_i^{\mathcal{B}}$-Replacement Schemes in W_1^i and TW_1^i

In this section we shall discuss various replacement schemes, also called collection or choice schemes by some authors, particularly in the context of second-order theories. These schemes allow third-order existential quantifiers to be moved past lower-order quantifiers, and are theorems of W_1^i and TW_1^i, as we shall show.

Since $W_1^1 \supset V(= \bigcup V^i)$, W_1^1 can $\Sigma_0^{\mathcal{B}}$-define all number- and string-valued functions of number and string arguments from the polynomial-time hierarchy. We may conservatively add symbols for such functions to W_1^i or TW_1^i and use them freely in all the axiom schemes. In particular, the string concatenation function $X \frown Y$ and pairing functions such as $\langle x, y\rangle$, $\langle X, Y\rangle$ and $\langle X, y\rangle$ may be added. For a third-order variable $\mathcal{X}$ define $\mathcal{X}^{[x]}(X) \equiv \mathcal{X}(\langle x, X\rangle)$ and $\mathcal{X}^{[X]}(Y) \equiv \mathcal{X}(\langle X, Y\rangle)$, which make $\mathcal{X}$ into an array, with rows indexed by number or strings respectively, each row of which is a third-order object. Note that this notation is to abbreviate a subformula, and not a superstring-valued function symbol; it applies only in the base theory that has no string- or superstring-valued function symbols. With this in mind, we can state the $\Sigma_1^{\mathcal{B}}$ replacement schemes for $\mathcal{L}_A^3$:

Definition 4.5.1 ($g\Sigma_1^{\mathcal{B}}$ Replacement Schemes)**.** *$g\Sigma_i^{\mathcal{B}}$-1REPL is:*

$$\forall x \leq y \exists \mathcal{X} \phi(x, y, \mathcal{X}) \leftrightarrow \exists \mathcal{X} \forall x \leq y \phi(x, y, \mathcal{X}^{[x]})$$

and $g\Sigma_i^{\mathcal{B}}$-2REPL is:

$$\forall X \leq y \exists \mathcal{X} \phi(X, y, \mathcal{X}) \leftrightarrow \exists \mathcal{X} \forall X \leq y \phi(X, y, \mathcal{X}^{[X]}),$$

where in each case ϕ is a (general) $\Sigma_1^{\mathcal{B}}$-formula which may have other free variables than those indicated.

Theorem 4.5.2. *The $g\Sigma_i^{\mathcal{B}}$ replacement schemes are theorems of W_1^i.*

Proof. Although the $g\Sigma_i^{\mathcal{B}}$-1REPL scheme has a simpler proof, it can also be proved in the same way as the $g\Sigma_i^{\mathcal{B}}$-2REPL scheme, so we include only a proof of the latter. This proof is analogous to and closely parallels Theorem 9.16 of Buss [3].

$\leftarrow$: In this direction, even for W_1^1 and $\phi(X, y, \mathcal{X}) \in g\Sigma_i^{\mathcal{B}}$, it is the case that

$$W_1^1 \vdash \exists \mathcal{X} \forall X \leq y \phi(X, y, \mathcal{X}^{[X]}) \supset \forall X \leq y \exists \mathcal{Z} \phi(X, y, \mathcal{Z}).$$

For a given X, $\Sigma_0^{\mathcal{B}}$-3COMP is used to obtain a superstring $\mathcal{Z}$ behaving like $\mathcal{X}^{[X]}$ up to a relevant bound depending on ϕ. Then by structural induction on ϕ we can construct a proof in W_1^1 that this $\mathcal{Z}$ satisfies $\phi(X, y, \mathcal{Z})$.

$\rightarrow$: First we show that W_1^1 proves this direction of $g\Sigma_1^{\mathcal{B}}$-2REPL, and we do so by structural induction on ϕ. The base case of the induction is when ϕ is $\Sigma_0^{\mathcal{B}}$. Let ψ be $\forall X \leq y \exists \mathcal{X} \phi(X, y, \mathcal{X})$. Let $\theta(c)$ be the formula

$$\forall X \leq (y \dot{-} c) \exists \mathcal{X} \forall Y \leq c \phi(X \frown Y, y, \mathcal{X}^{[Y]}).$$

$\theta(0)$ is a simple consequence of ψ, and $W_1^1 \vdash \psi \wedge \theta(c) \supset \theta(c+1)$ by use of $\Sigma_0^{\mathcal{B}}$-3COMP to combine two third-order objects (coding the two arrays of third-order objects for all strings of length smaller than y starting with $X \frown 0$ and $X \frown 1$ respectively) into one third-order object coding the array for all strings of length smaller than y starting with X. Thus $W_1^1 \vdash \psi \supset \theta(y)$ by $\forall^2\Sigma_1^{\mathcal{B}}$-IND, and clearly $W_1^1 \vdash \theta(y) \supset \exists\mathcal{X}\forall X \leq y\phi(X, y, \mathcal{X}^{[X]})$.

Now let $k > 0$ and assume the present theorem holds for every member of $g\Sigma_1^{\mathcal{B}}$ with fewer than k third-order quantifiers. Let $\phi \in g\Sigma_1^{\mathcal{B}}$ have exactly k third-order quantifiers and assume without loss of generality that ϕ is in prenex normal form. (Every formula is provably in W_1^1 equivalent to one in prenex normal form). Then every third-order quantifier in ϕ is existential, and $\phi(X, y, \mathcal{X})$ is of the form $Q_1\tilde{a_1}...Q_n\tilde{a_n}\exists\mathcal{Z}\psi(\tilde{a_1}, ..., \tilde{a_n}, \mathcal{Z}, X, y, \mathcal{X})$ for some n and ψ with $k-1$ existential third-order quantifiers. Each Q_i is a bounded first- or second-order quantifier and the corresponding $\tilde{a_i}$ is a variable of the appropriate sort. By several applications of the inductive hypothesis we prove

$$Q_1\tilde{a_1}...Q_n\tilde{a_n}\exists\mathcal{Z}\psi(\tilde{a_1}, ..., \tilde{a_n}, \mathcal{Z}, X, y, \mathcal{X}) \supset \exists\mathcal{Z}Q_1\tilde{a_1}...Q_n\tilde{a_n}\psi(\tilde{a_1}, ..., \tilde{a_n}, \mathcal{Z}^{[\tilde{a_1}]...[\tilde{a_n}]}, X, y, \mathcal{X}). \tag{4.5.1}$$

The inductive hypothesis is not needed for those Q_i which are existential, nor in that case need we add $[\tilde{a_i}]$ to the formula on the right of the equivalence, yet it is harmless and simplifies matters to do so.

Now with $\Sigma_0^{\mathcal{B}}$-3COMP we can prove

$$\exists\mathcal{X}\exists\mathcal{Z}Q_1\tilde{a_1}...Q_n\tilde{a_n}\psi(\tilde{a_1}, ..., \tilde{a_n}, \mathcal{Z}^{[\tilde{a_1}]...[\tilde{a_n}]}, X, y, \mathcal{X}) \supset$$
$$\exists\mathcal{Z}Q_1\tilde{a_1}...Q_n\tilde{a_n}\psi(\tilde{a_1}, ..., \tilde{a_n}, \mathcal{Z}^{[1][\tilde{a_1}]...[\tilde{a_n}]}, X, y, \mathcal{Z}^{[2]}) \tag{4.5.2}$$

and thus piecing together implications 4.5.1 and 4.5.2 we obtain

$$\forall X \leq y\exists\mathcal{X}\phi(X, y, \mathcal{X}) \supset \forall X \leq y\exists\mathcal{Z}Q_1\tilde{a_1}...Q_n\tilde{a_n}\psi(\tilde{a_1}, ..., \tilde{a_n}, \mathcal{Z}^{[1][\tilde{a_1}]...[\tilde{a_n}]}, X, y, \mathcal{Z}^{[2]}).$$

We may now appeal to the inductive hypothesis once more and apply the current theorem to the right-hand side of the previous implication, which results in

$$\forall X \leq y\exists\mathcal{X}\phi(X, y, \mathcal{X}) \supset \exists\mathcal{Z}\forall X \leq yQ_1\tilde{a_1}...Q_n\tilde{a_n}\psi(\tilde{a_1}, ..., \tilde{a_n}, \mathcal{Z}^{[X][1][\tilde{a_1}]...[\tilde{a_n}]}, X, y, \mathcal{Z}^{[X][2]}).$$

By applying $\Sigma_0^{\mathcal{B}}$-3COMP we can separate in two along the second "co-ordinate" the object $\mathcal{Z}$, quantified in the right-hand side:

$$\forall X \leq y \exists \mathcal{X} \phi(X, y, \mathcal{X}) \supset \exists \mathcal{X} \exists \mathcal{Z} \forall X \leq y Q_1 \tilde{a_1} ... Q_n \tilde{a_n} \psi(\tilde{a_1}, ..., \tilde{a_n}, \mathcal{Z}^{[X][\tilde{a_1}]...[\tilde{a_n}]}, X, y, \mathcal{X}^{[X]}).$$

The formula

$$\exists \mathcal{X} \forall X \leq y Q_1 \tilde{a_1} ... Q_n \tilde{a_n} \exists \mathcal{Z} \psi(\tilde{a_1}, ..., \tilde{a_n}, \mathcal{Z}, X, y, \mathcal{X}^{[X]})$$

is a logical consequence of the right-hand side of the previous implication and so we have proved

$$\forall X \leq y \exists \mathcal{X} \phi(X, y, \mathcal{X}) \supset \exists \mathcal{X} \forall X \leq y \phi(X, y, \mathcal{X}^{[X]}),$$

as required.

Now at this point we know that $W_1^1 \vdash g\Sigma_1^{\mathcal{B}}$-2REPL. Inductively, if $W_1^{i-1} \vdash g\Sigma_{i-1}^{\mathcal{B}}$-2REPL, we can show that $W_1^i \vdash g\Sigma_i^{\mathcal{B}}$-2REPL as follows: Let $\phi \in g\Sigma_i^{\mathcal{B}}$ be in prenex form. By the inductive hypothesis, the maximal subformula ψ of ϕ that is $g\Sigma_{i-1}^{\mathcal{B}}$ is equivalent to some (strict) $\Sigma_{i-1}^{\mathcal{B}}$-formula ψ'. Now as in the base case for W_1^1, we must perform a structural induction based on the number of remaining third-order quantifiers in ϕ. The base case will apply $\forall^2\Sigma_i^{\mathcal{B}}$-IND to address a single additional third-order quantifier, and the induction step will merge two third-order quantifiers in order to apply the induction hypothesis.

□

The schemes are also provable in TW_1^i:

Theorem 4.5.3. *The $g\Sigma_i^{\mathcal{B}}$ replacement schemes are theorems of TW_1^i.*

Proof Sketch. The proof is as for the previous theorem, except that where $\forall^2\Sigma_i^{\mathcal{B}}$-IND was formerly used, we instead construct the $\mathcal{X}$ on the right-hand side by an application of $\Sigma_i^{\mathcal{B}}$-SIND. Since the length of the induction is now exponentially longer, we have no need of Buss's trick with the leading second-order universal quantifier. □

The following is an immediate, useful corollary:

Corollary 4.5.4. *Let* $\phi \in g\Sigma_i^{\mathcal{B}}$. *Then there exists* $\psi \in$ *(strict)*$\Sigma_i^{\mathcal{B}}$ *such that* $W_1^i \vdash \phi \leftrightarrow \psi$ *and* $TW_1^i \vdash \phi \leftrightarrow \psi$.

Chapter 5

Definability in the Theories

In this chapter we prove a number of definability results showing that functions from certain complexity classes are definable in our theories. These theorems rely heavily on standard techniques for arithmetization of Turing machine configurations and computations; expressibility of predicates such as a "next relation"; and standard functions giving the output of a computation or the initial configuration of a machine.

5.1 Definability in W_1^i and $\widehat{W_1^1}$

We know that W_1^1 can $\Sigma_0^{\mathcal{B}}$-define all string-valued functions (of string variables) from the polynomial-time hierarchy. In fact, W_1^1 can $\Sigma_1^{\mathcal{B}}$-define all string functions computable in polynomial space, and more generally, W_1^i $\Sigma_i^{\mathcal{B}}$-defines all functions from $(\text{FPSPACE}^{(\Sigma_{i-1}^{exp})^\diamond})^+$, which in the case of $i = 1$ is simply the third-order generalization FPSPACE^+ of PSPACE.

Theorem 5.1.1. *Let $i \geq 1$ and $\tilde{f} \in (FPSPACE^{(\Sigma_{i-1}^{exp})^\diamond})^+$. Then $\tilde{f}$ is $\Sigma_i^{\mathcal{B}}$-definable in W_1^i.*

Proof. First, consider a superstring-valued function $\mathcal{F}(\overline{\tilde{x}})$ $\in(\text{FPSPACE}^{(\Sigma_{i-1}^{exp})^\diamond})^+$. The third-order predicate $f_{\mathcal{F}}(\overline{\tilde{x}}, Y) \equiv \mathcal{F}(\overline{\tilde{x}})(Y)$ is clearly in the same class. Furthermore, if $f_{\mathcal{F}}$ is $\Sigma_i^{\mathcal{B}}$-definable in W_1^i, then the superstring-valued function $\mathcal{F}'$ whose output codes the outputs of $f_{\mathcal{F}}$ for values of Y up to a bound t is also (length 2^t-) $\Sigma_i^{\mathcal{B}}$-definable, by essentially the same argument as the replacement schemes: By induction on y up to t, W_1^i proves that for each string $Z \leq y$ there is a (boundedly) unique superstring $\mathcal{X}$ coding the values of $f_{\mathcal{F}}(\overline{\tilde{x}}, Y)$ for the Y having Z as prefix. At the end of the induction, therefore, the superstring $\mathcal{X}$ codes all the desired values of $f_{\mathcal{F}}$. If t is a true bound for $\mathcal{F}$, then this means that $\mathcal{F}$ is $\Sigma_i^{\mathcal{B}}$-definable in W_1^i.

An even simpler argument shows that if string-valued functions are definable, then so are number-valued functions. We therefore focus on the case of string-valued functions.

Let $F \in$(FPSPACE$^{(\Sigma_{i-1}^{exp})^{\diamond}}$)$^+ \cap \mathbb{E}^2$, and M be a third-order-oracle PSPACE Turing machine such that $M^{\Sigma_{i-1}^{\mathcal{B}}}$ computes F and $s(\overline{\tilde{x}})$ be a (number) term bounding the space used by M (including M's output tape) on input $\overline{\tilde{x}}$ (and thus also bounding the logarithm of the running time). Let $\phi_M(\mathcal{W}, \overline{\mathcal{X}}, Y, Z, l)$ state that $\mathcal{W}$ is a computation of $M^{\Sigma_{i-1}^{\mathcal{B}}}$ of length 2^l steps, with initial configuration coded by Y and final configuration coded by Z, where $\overline{\mathcal{X}}$ are the (read-only) superstring inputs to M. $\mathcal{W}$ is stored as an array of configurations, indexed by configuration number expressed as a string, and for the sake of simplicity ϕ_M enforces that all configurations are the same size. $\phi_M(\mathcal{W}, \overline{\mathcal{X}}, Y, Z, l)$ states that for each configuration number smaller than 2^l (bounded second order universal quantifier) the corresponding configuration is valid and results from the previous one by one step of M ($g\Sigma_i^B$ subformula incorporating a $\Sigma_{i-1}^{\mathcal{B}}$ subformula for the oracle). Thus ϕ_M is $g\Sigma_i^{\mathcal{B}}$. For concreteness in what follows, we shall reason in the sequent formulation $\mathrm{LK}^3 - W_1^i$ of W_1^i.

Even W_1^1 can clearly then prove

$$\forall X \leq |S| \exists \mathcal{W} \exists Y \leq |X| \phi_M(\mathcal{W}, \overline{\mathcal{X}}, X, Y, 0);$$

Either the $\Sigma_{i-1}^{\mathcal{B}}$ oracle (if queried) accepts or not, and in either case computing the next configuration is straightforward.

Now, W_1^i proves

$$\phi_M(\mathcal{Y}, \overline{\mathcal{X}}, A, B, l), \phi_M(\mathcal{Z}, \overline{\mathcal{X}}, B, C, l) \longrightarrow \exists \mathcal{W} \exists Y \leq |A| \phi_M(\mathcal{W}, \overline{\mathcal{X}} A, Y, l+1),$$

since $\Sigma_0^{\mathcal{B}}$-3COMP can be used to produce the third-order object $\mathcal{W}$ which consists of $\mathcal{Y}$ and $\mathcal{Z}$ spliced together, and even W_1^1 can subsequently prove that such $\mathcal{X}$ satisfies ϕ_M as shown. Now being careful of the order in which we do so, we may introduce quantifiers in this sequent as follows, using the fact that all configurations in a computation are the same size: First

$$\phi_M(\mathcal{Y}, \overline{\mathcal{X}}, A, B, l), \exists \mathcal{W} \exists Y \leq |B| \phi_M(\mathcal{W}, \overline{\mathcal{X}}, B, Y, l) \longrightarrow \\ \exists \mathcal{W} \exists Y \leq |A| \phi_M(\mathcal{W}, \overline{\mathcal{X}}, A, Y, l+1).$$

Adding a hypothesis to the succedent we obtain

$$\phi_M(\mathcal{Y}, \overline{\mathcal{X}}, A, B, l), \exists \mathcal{W} \exists Y \leq |B| \phi_M(\mathcal{W}, \overline{\mathcal{X}}, B, Y, l) \longrightarrow \\ |A| \leq |S| \supset \exists \mathcal{W} \exists Y \leq |A| \phi_M(\mathcal{W}, \overline{\mathcal{X}}, A, Y, l+1),$$

then by reasoning about sizes of the configurations we may add a similar hypothesis to the second formula in the antecedent:

$$\phi_M(\mathcal{Y}, \overline{\mathcal{X}}, A, B, l), |B| \leq |S| \supset \exists \mathcal{W} \exists Y \leq |B| \phi_M(\mathcal{W}, \overline{\mathcal{X}}, B, Y, l) \longrightarrow \\ |A| \leq |S| \supset \exists \mathcal{W} \exists Y \leq |A| \phi_M(\mathcal{W}, \overline{\mathcal{X}}, A, Y, l+1),$$

and then introduce a quantifier like so:

$$\phi_M(\mathcal{Y}, \overline{\mathcal{X}}, A, B, l), \forall X \leq |S| \exists \mathcal{W} \exists Y \leq |X| \phi_M(\mathcal{W}, \overline{\mathcal{X}}, X, Y, l) \longrightarrow \\ |A| \leq |S| \supset \exists \mathcal{W} \exists Y \leq |A| \phi_M(\mathcal{W}, \overline{\mathcal{X}}, A, Y, l+1).$$

Then similar reasoning with the first formula in the antecedent yields

$$\forall X \leq |S| \exists \mathcal{W} \exists Y \leq |X| \phi_M(\mathcal{W}, \overline{\mathcal{X}}, X, Y, l), \forall X \leq |S| \exists \mathcal{W} \exists Y \leq |X| \phi_M(\mathcal{W}, \overline{\mathcal{X}}, X, Y, l) \\ \longrightarrow |A| \leq |S| \supset \exists \mathcal{W} \exists Y \leq |A| \phi_M(\mathcal{W}, \overline{\mathcal{X}}, A, Y, l+1).$$

Contraction and the introduction of a final quantifier in the succedent yields a sequent suitable for applying induction to:

$$\forall X \leq |S| \exists \mathcal{W} \exists Y \leq |X| \phi_M(\mathcal{W}, \overline{\mathcal{X}}, X, Y, l) \longrightarrow \\ \forall X \leq |S| \exists \mathcal{W} \exists Y \leq |X| \phi_M(\mathcal{W}, \overline{\mathcal{X}}, X, Y, l+1),$$

and $g\Sigma_i^{\mathcal{B}}$-IND produces

$$\longrightarrow \forall X \leq |S| \exists \mathcal{W} \exists Y \leq |X| \phi_M(\mathcal{W}, \overline{\mathcal{X}}, X, Y, |X|).$$

Now it is easy to see that W_1^1 proves

$$\exists Y \exists \mathcal{W} \exists Z \leq s(|\overline{\tilde{x}}|)(\phi_M(\mathcal{W}, \overline{\mathcal{X}}, \mathrm{I}nit_M(\overline{\tilde{x}}), Z, s(\overline{\tilde{x}})) \wedge \mathrm{O}ut_M(Z, Y))$$

for a suitable function symbol $\mathrm{I}nit_M$ and formula $\mathrm{O}ut_M$. Thus the existence part of the definability is obtained. The fact that the output of function F satisfies this formula is clear given the definition of ϕ_M from the Turing machine M computing f.

Finally, uniqueness is proved as follows: Firstly, define $\psi(k)$ to be the formula

$$\forall A \leq s(\overline{\tilde{x}}) \forall B \leq s(\overline{\tilde{x}}) \forall C \leq s(\overline{\tilde{x}}) \forall Z \leq s(\overline{\tilde{x}}) \\ (\phi_M(sub(\mathcal{W}, Z, k), \overline{\mathcal{X}}, A, B, k) \wedge \phi_M(sub(\mathcal{Y}, Z, k), \overline{\mathcal{X}}, A, C, k) \supset B = C),$$

where $sub(\mathcal{W}, Z, l)$ is a suitably defined functional which gives the subcomputation of $\mathcal{W}$ starting with configuration number encoded by Z and continuing for 2^l steps. In other words, $\mathrm{sub}(\mathcal{W}, Z, l)(X)$ abbreviates a subformula that evaluates to the Xth bit of this subcomputation.

Now, $\psi(0)$ is provable in W_1^1 since the next configuration of an oracle Turing machine is computable in linear time (with the oracle), and thus is definable even in V^1. $\psi(l) \supset \psi(l+1)$ is immediate, and so by ($\Sigma_i^{\mathcal{B}}$-)induction, $\psi(s(\overline{\tilde{x}}))$, from which uniqueness follows.

$\square$

Now we show that the case of $i = 1$ of the above theorem is true for $\widehat{W_1^1}$. It is unfortunately not clear how this result could be generalized as in the previous theorem; it would seem that a stronger form of comprehension could be needed to construct the initial array of all 1-step computations. This theorem is not directly analogous to any existing theorem (for example, for U_2^1) and so is somewhat interesting.

Theorem 5.1.2. *Let $\tilde{f} \in$ FPSPACE$^+$. Then $\tilde{f}$ is strict-$\Sigma_1^{\mathcal{B}}$-definable in $\widehat{W_1^1}$.*

Proof. As in the previous theorem, we may restrict our attention to the case of a string-valued function F, but we must use a slightly different argument. As before, for a superstring-valued function $\mathcal{F}$ we can define the associated predicate $f_{\mathcal{F}}(Y)$. However, for the same reason that $\widehat{W_1^1}$ does not obviously prove the replacement schemes, it does not obviously define $\mathcal{F}'$, outputting a superstring encoding all the values of $f_{\mathcal{F}}(Y)$ for Y up to a true bound t on $\mathcal{F}$. The solution to this problem is as follows: the argument below concerning string-valued functions shows in particular that $\widehat{W_1^1}$ proves the existence of a computation of a given FPSPACE$^+$ machine. Consider a machine M that computes the values of $f_{\mathcal{F}}(Y)$ for each Y up to the bound t, in such a way that each subcomputation is of equal length, and thus that the values of $f_{\mathcal{F}}(Y)$ are in easily-computed locations. Now given this computation and using $\Sigma_0^{\mathcal{B}}$-3COMP, $\widehat{W_1^1}$ can prove the existence and (bounded) uniqueness of a superstring consisting of exactly the crucial bits $f_{\mathcal{F}}(Y)$ of this computation, and this superstring is of course the value of $\mathcal{F}$.

Now, let $F \in$FPSPACE$^+ \cap \mathbb{E}^2$ and let ϕ_M be as in the previous theorem for a machine M computing F. Let

$$\phi'_M(\mathcal{W}, \overline{\mathcal{X}}, i, S) \equiv \forall X \leq |S| \exists Y \leq |S| \phi_M(\mathcal{W}^{[X]}, \overline{\mathcal{X}}, X, Y, i).$$

This $\Sigma_0^{\mathcal{B}}$ formula expresses that $\mathcal{W}$ simultaneously encodes computations of M of length 2^i from every starting configuration of length $|S|$ to some ending configuration.

We now reason in LK$^3 - \widehat{W_1^1}$ and seek to proceed by induction on i. A $\mathcal{W}$ provably satisfying $\phi'_M(\mathcal{W}, \overline{\mathcal{X}}, 0, S)$ is obtained by comprehension on a $\Sigma_0^{\mathcal{B}}$ formula (as computing a requested bit of $\mathcal{W}$ is even in polynomial time). That it satisfies $\phi'_M(\mathcal{W}, \overline{\mathcal{X}}, 0, S)$ follows easily so we have

$$\exists \mathcal{W} \phi'_M(\mathcal{W}, \overline{\mathcal{X}}, 0, S).$$

Now,

$$\exists \mathcal{W} \phi'_M(\mathcal{W}, \overline{\mathcal{X}}, i, S) \longrightarrow \exists \mathcal{Y} \phi'(\mathcal{Y}, \overline{\mathcal{X}}, i+1, S)$$

is proved by one application of $\Sigma_0^{\mathcal{B}}$-3COMP: given a $\mathcal{W}$ satisfying $\phi'_M(\mathcal{W}, \overline{\mathcal{X}}, i, S)$, a $\mathcal{Y}$ satisfying $\phi_M(\mathcal{Y}, \overline{\mathcal{X}}, i+1, S)$ is defined by a $\Sigma_0^{\mathcal{B}}(\mathcal{W})$-formula that for any requested initial configuration splices together the two relevant computations from the given $\mathcal{W}$.

The rest of the proof is analogous to that of the previous theorem. □

5.2 Definability in TW_1^i and TTW_1^0

In this section we prove definability results for TW_1^i and TTW_1^0.

Theorem 5.2.1. *Let $i \geq 1$ and $\tilde{f} \in (FEXP^{(\Sigma_{i-1}^{exp})^\diamond})^+$. Then $\tilde{f}$ is $\Sigma_i^{\mathcal{B}}$-definable in TW_1^i.*

Proof. As in Theorem 5.1.1 we need only consider string-valued $F(\overline{\tilde{x}})$. Let M be a third-order oracle EXP Turing machine computing F with access to a $\Sigma_{i-1}^{\mathcal{B}}$ oracle and $t(\overline{\tilde{x}})$ be a number term bounding the logarithm of the running time of M on input $\overline{\tilde{x}}$. Let $\phi_M(\mathcal{W}, \overline{\tilde{x}}, L)$ state that $\mathcal{W}$ is a computation of $M^{\Sigma_{i-1}^{\mathcal{B}}}(\overline{\tilde{x}})$ of length L steps (coding a number). Again, $\mathcal{W}$ is an array of configurations (although now they must be themselves coded as superstrings as they may be of exponential size) and as before, $\phi_M \in g\Sigma_i^{\mathcal{B}}$. Since configurations are coded by superstrings, ϕ_M actually checks only that an initial segment of each row of $\mathcal{W}$ codes the relevant configuration.

As before,

$$W_1^1 \vdash \exists \mathcal{W} \phi_M(\mathcal{W}, \overline{\tilde{x}}, \mathrm{One}),$$

where One is a constant for the string coding 1.

$$\exists \mathcal{W} \phi_M(\mathcal{W}, \overline{\tilde{x}}, L) \longrightarrow \exists \mathcal{W} \phi_M(\mathcal{W}, \overline{\tilde{x}}, L+1)$$

(for $L+1$ a string successor function) is provable in TW_1^i by applying $\Sigma_0^{\mathcal{B}}$-3COMP for the two cases of whether or not the $\Sigma_{i-1}^{\mathcal{B}}$ oracle (if relevant) accepts. Thus by $g\Sigma_i^{\mathcal{B}}$-SIND,

$$TW_1^i \vdash \exists \mathcal{W} \phi_M(\mathcal{W}, \overline{\overline{x}}, 2^{t(\overline{\overline{x}})}).$$

Therefore

$$TW_1^i \vdash \exists Y \exists \mathcal{W}(\phi_M(\mathcal{W}, \overline{\overline{x}}, 2^{t(\overline{\overline{x}})}) \wedge \mathrm{Out}_M(\mathcal{W}, \overline{\overline{x}}, Y))$$

for a suitable formula Out_M.

Uniqueness is proved analogously to in Theorem 5.1.1, except that $g\Sigma_i^{\mathcal{B}}$-SIND is used to prove **bounded** equality of the configurations in any two computations $\mathcal{W}$ and $\mathcal{W}'$, and from this follows uniqueness of the string output. □

Now we show that all functions from FEXP$^+$ are $\Sigma_1^{\mathcal{B}}$-definable in TTW_1^0. This is because TTW_1^0 proves $\Sigma_0^{\mathcal{B}}$-superstring-recursion:

$$\exists \mathcal{X} \phi^{\mathrm{rec}}(S, \mathcal{X}),$$

where $\phi \in \Sigma_0^{\mathcal{B}}$, and

$$\phi^{\mathrm{rec}}(S, \mathcal{X}) \equiv \forall Y \leq |S|(L_2(Y, S) \supset (\mathcal{X}(Y) \leftrightarrow \phi(Y, \mathcal{X}^{<Y}))),$$

where $\mathcal{X}^{<Y}$ is a chop function.

Lemma 5.2.2. *TTW_1^0 proves the $\Sigma_0^{\mathcal{B}}$-superstring-recursion scheme*

Proof. Even TW_1^0 can prove (by induction on strings Y up to length x) that $\phi^{\mathrm{rec}}(S, \mathcal{X}) \wedge \phi^{\mathrm{rec}}(S, \mathcal{Z})$ implies the bits of $\mathcal{X}$ and $\mathcal{Z}$ are equal up to bit number S. Now, following the analogous exposition from [27] for TV^0, define

$$\phi^{\mathrm{lessrec}}(S, \mathcal{X}) \equiv \phi^{\mathrm{rec}}(S, \mathcal{X}) \vee [\exists Y \leq |S|(L_2(Y, S) \wedge \phi^{\mathrm{rec}}(Y, \mathcal{X}) \wedge \neg \mathcal{X}(Y) \wedge \phi(Y, \mathcal{X}^{<Y}))],$$

stating that either $\phi^{\mathrm{rec}}(S, \mathcal{X})$, or that some prefix of $\mathcal{X}$ is correct up to position Y, where $\phi(Y, \mathcal{X}^{<Y})$, yet $\neg \mathcal{X}(Y)$. In other words, if $Rev(S, \mathcal{X})$ is a function symbol reversing the

order of the first S bits of $\mathcal{X}$ (putting the most significant bits first), then $Rev(S, \mathcal{X})$ is "less than" the unique string $\mathcal{Y}$ satisfying $\phi^{\text{lessrec}}(S, Rev(S, \mathcal{Y}))$.

Reasoning in TTW_1^0, $\phi^{\text{lessrec}}(S, \mathcal{Z})$ holds, where $\mathcal{Z}$ is a null object, by induction on S. Now define $\mathcal{W}$ by $\forall X \leq |S|[\mathcal{W}(X) \leftrightarrow 1]$. $\mathcal{W}$ is an "all-ones" superstring, and if $\phi^{\text{lessrec}}(S, \mathcal{W})$, then $\phi^{\text{rec}}(S, \mathcal{W})$, which would complete the proof. Therefore assume $\neg\phi^{\text{lessrec}}(S, \mathcal{W})$. By $\Sigma_0^{\mathcal{B}}$-SSIND for $\phi^{\text{lessrec}}(S, Rev(S, \mathcal{X}))$, (since Rev fixes both $\mathcal{Z}$ and $\mathcal{W}$ above),

$$\exists \mathcal{X} \exists \mathcal{Y} \phi^{\text{lessrec}}(S, Rev(S, \mathcal{X})) \wedge S_3(\mathcal{X}, \mathcal{Y}, S) \wedge \neg\phi^{\text{lessrec}}(S, Rev(S, \mathcal{Y})).$$

Now TW_1^0 proves by induction on S that $\phi^{\text{rec}}(S, Rev(S, \mathcal{X}))$. □

Now we show how to define EXP-time computations using $\Sigma_0^{\mathcal{B}}$-superstring-recursion:

Theorem 5.2.3. *Let $\tilde{f} \in FEXP^+$. Then $\tilde{f}$ is strict-$\Sigma_1^{\mathcal{B}}$-definable in TTW_1^0.*

Proof Sketch. As in Theorem 5.1.2, the case of a superstring-valued function is a simple modification of the argument below: first define a function outputting the computation of a machine computing each bit of the superstring output one at a time, and then compose with a function condensing out the bits of the superstring output.

Thus let F be string-valued and as described and M an exponential-time Turing machine computing F, with $s(\overline{\tilde{x}})$ a number term bounding the output size of M on input $\overline{\tilde{x}}$ and $t(\overline{\tilde{x}})$ bounding the logarithm of the run-time. We describe a formula $\phi_M(\overline{\tilde{x}}, Y, \mathcal{X})$ suitable for applying the above recursion scheme to. The arguments $\overline{\tilde{x}}$ denotes the inputs to M, and Y and $\mathcal{X}$ are as in the recursion scheme. $\phi_M(\overline{\tilde{x}}, Y, \mathcal{X}) \equiv \phi_M^1(\overline{\tilde{x}}, Y, \mathcal{X}) \vee \phi_M^2(\overline{\tilde{x}}, Y, \mathcal{X})$.

The disjunct ϕ_M^1 evaluates to the appropriate bit of the initial configuration of M on input $\overline{\tilde{x}}$ if Y is small enough.

The other disjunct ϕ_M^2 is as follows: For every two positions W_1, W_2 smaller than Y, if $L_2(W_1, W_2)$ and W_1 and W_2 point to the start of configurations in $\mathcal{X}$, and there is no W_3

between W_1 and W_2 or between W_2 and Y also pointing to the start of a configuration, then M would correctly compute bit Y of the latter configuration to be 1. Bit Y is a function of: possibly one bit of the read-only superstring inputs among $\bar{\bar{x}}$ (in case of a read operation); possibly polynomially many bits of $\mathcal{X}$ following W_2 representing the query tape for the superstring inputs; and finally a constant number of bits of $\mathcal{X}$ preceding Y and preceding $W_1 +_2 (Y -_2 W_2)$, where '$+_2$' and '$-_2$' are intended to represent arithmetic on strings.

ϕ_M as outlined is then clearly $\Sigma_0^{\mathcal{B}}$ (in fact, even Π_2^B).

Now, applying the $\Sigma_0^{\mathcal{B}}$-superstring-recursion, $TTW_1^0 \vdash \exists\mathcal{X}\phi_M^{\text{rec}}(X, Y, \mathcal{X})$. For a suitable formula Out_M, $TTW_1^0 \vdash \forall X \exists Y(\exists\mathcal{X}(\phi_M^{\text{rec}}(X, (2^{t(X)})^2, \mathcal{X}) \wedge Out_M(\mathcal{X}, (2^{t(X)})^2, Y)))$. Point 2 of the definability is clear, and uniqueness follows directly from the (bounded) uniqueness of superstrings satisfying ϕ_M^{rec}.

□

5.3 Definability in HW_1^0

In this final section we show that HW_1^0 $\Sigma_1^{\mathcal{B}}$-defines the PSPACE functions:

Theorem 5.3.1. *Let $\tilde{f} \in FPSPACE^+$. Then $\tilde{f}$ is strict-$\Sigma_1^{\mathcal{B}}$-definable in HW_1^0.*

Proof. We begin by addressing the case for a string-valued function F.

Therefore let F be as described and M a polynomial-space Turing machine computing F, with $s(\bar{\bar{x}})$ a number term bounding the space used by M on input $\bar{\bar{x}}$. The idea now is to use a superstring $\mathcal{X}$ to encode a sequence of adjacency matrices. The ith matrix will indicate, for every pair of configurations of $M(\bar{\bar{x}})$, if one yields the other in time at most 2^i. Furthermore, these matrices will alternate with unused space, so that the halfrecursion scheme can be applied. Each matrix will be of size exponential in $\bar{\bar{x}}$, and the constructed superstring containing will be doubled in length each time a new matrix is added, in total a polynomial number of times.

To that end we now describe a formula $\phi_M(\overline{\overline{x}}, Y, \mathcal{X})$ suitable for application of the halfrecursion scheme. $\overline{\overline{x}}$ is the input to M and Y and $\mathcal{X}$ are as in the recursion scheme, and as before, $\phi_M(\overline{\overline{x}}, Y, \mathcal{X}) \equiv \phi^1_M(\overline{\overline{x}}, Y, \mathcal{X}) \vee \phi^2_M(\overline{\overline{x}}, Y, \mathcal{X})$.

The disjunct ϕ^1_M is $|Y| = 2s(\overline{\overline{x}}) \wedge \exists Z, W(Y = Z \frown W \wedge (\mathrm{N}ext_M(\overline{\overline{x}}, Z, W) \wedge Z = W))$. This subformula ignores $\mathcal{X}$ and directly computes bit Y, if Y is the right length to indicate a pair of configurations.

The other disjunct ϕ^2_M is as follows:

$$\exists A, B, C \leq |Y|(2|A| = 2|B| = 2|C| = |Y| - 2 \wedge Y = 0^2 \frown A \frown C \\ \wedge \mathcal{X}(A \frown B) \wedge \mathcal{X}(B \frown C)).$$

This subformula verifies that Y is 00 followed by some pair of equal-length strings, and furthermore that 2 steps in the previous adjacency matrix in $\mathcal{X}$ yield the transition coded by Y.

So ϕ_M is $\Sigma^{\mathcal{B}}_0$ and $HW^0_1 \vdash \exists \mathcal{X} \phi^{\mathrm{hrc}}_m(\overline{\overline{x}}, Y, \mathcal{X})$. Let $\mathrm{O}ut_M$ be a suitable formula extracting the output from the least accepting configuration reachable from the starting configuration of $F(\overline{\overline{x}})$, as coded in $\mathcal{X}$. Then $\mathrm{O}ut_M$ is clearly $\Sigma^{\mathcal{B}}_0$ and $HW^0_1 \vdash \forall \overline{\overline{x}} \exists Y(\exists \mathcal{X}(\phi^{\mathrm{hrc}}_M(\overline{\overline{x}}, 2^{6s(\overline{\overline{x}})}, \mathcal{X}) \wedge \mathrm{Out}_M(\mathcal{X}, \overline{\overline{x}}, Y)))$, satisfying the existence part of the definability. Point 2 is clear and uniqueness is as in the previous theorem, using the provable (bounded) uniqueness of superstrings satisfying the recursion schemes.

Now, the case of a number-valued function follows directly. For a superstring-valued function $\mathcal{F}$, the above argument shows that HW^0_1 proves the existence of the adjacency matrix of a machine M computing the predicate $f_{\mathcal{F}}(\overline{\overline{x}}, Y) \equiv \mathcal{F}(\overline{\overline{x}})(Y)$. Since initial configurations of this machine for varying values of Y are $\Sigma^{\mathcal{B}}_0$-definable in W^0_1, a single application of $\Sigma^{\mathcal{B}}_0$-3COMP can construct, from the adjacency matrix of the machine, a superstring coding the values of $f_{\mathcal{F}}$ for all Y up to the desired bound; this superstring is of course the value $\mathcal{F}(\overline{\overline{x}})$. (Bounded) uniqueness is again straightforward. $\square$

Chapter 6

A Universal Conservative Extension of HW_1^0

In this chapter we define and develop $\overline{HW_1^0}$, a universal theory intended to be a conservative extension of HW_1^0. We loosely follow similar constructions of universal theories for P, NL and so on from [27] and [44, 45]; however, our situation is considerably more complex as we have an additional sort (superstrings), and furthermore objects of that sort are unbounded.

We start with several additional function symbols beyond those provided in $\mathcal{L}_A^3$: recall the function f_{SE}, with open defining axioms SE' and SE", used as open replacements for the string extensionality axiom SE. B14' and B14" are open replacements for B14 defining the function *pd*. The superstring stretch function $\varsigma(a, b, \mathcal{X})$ is intended to return a superstring with the initial 2^{a+b} bits fixed such that the first 2^a of them agree with the input $\mathcal{X}$, and the remainder are zeroes. The open defining axiom is:

$$|Y| \leq a + b \supset (\varsigma(a, b, \mathcal{X})(Y) \leftrightarrow |Y| \leq a \wedge \mathcal{X}(Y)).$$

All these functions are $\Sigma_0^{\mathcal{B}}$-definable in W_1^0 (length s^{a+b}-definable in the case of ς).

The open theory $\overline{HW_1^0}$ we define below defines a succession of function symbols $\mathcal{L}_{PS}$ inductively from previous ones, and specifies a set of defining axioms for each one. This requires more care than in analogous constructions because of the unbounded nature of our third-order objects, and the limited sense in which superstring-valued functions can be definable. For this reason, we shall associate with each function and predicate symbol a polynomial (i.e., term in $\mathcal{L}_A^3$) which will be an upper-bound on its sensitivity to its third-order arguments, as a function of its other arguments. Each function symbol will be provably in HW_1^0 insensitive to its third-order arguments beyond this bound, in the sense of definition 4.3.4. To function symbols we will additionally associate a polynomial ($\mathcal{L}_A^3$-term) bounding the output as a function of the number and string arguments. Each number- or string-valued function symbol will be definable in HW_1^0 and provably bounded by its associated polynomial t, while superstring-valued function symbols will be length-2^t-definable. These terms will be explicitly written into the names of all function symbols defined below.

The following definition shows how to extend these sensitivity and bounding polynomials to certain terms and open formulas. It also identifies a class of open formulas called **permissible formulas** that are constructed with sufficient interleavings of the superstring stretch function to ensure their value is well defined, and hence suitable for use in defining new function symbols:

Definition 6.0.2. *a) The bounding polynomials for 0, 1, $x+y$, $x*y$ and $|X|$ are respectively 0, 1, $x+y$, $x*y$ and $|X|$. The sensitivity polynomials for these function symbols are all 0.*

b) The bounding polynomials of $pd(x)$ and $f_{SE}(X,Y)$ are x and $|X|$ respectively, and sensitivity polynomials for both are 0.

c) The superstring stretch function $\varsigma(a,b,\mathcal{X})$ has sensitivity a and bound $a+b$.

d) If $\tilde{f}$ is a function symbol (of any type) with sensitivity s and bound t, $\mathcal{R}_1,...,\mathcal{R}_k$ are superstring terms with sensitivity $u_1,...,u_k$ and bound $v_1,...,v_k$, and finally $\tilde{h}_1,...,\tilde{h}_j$ are number or string terms with sensitivity $p_1,...,p_j$ and bounds $q_1,...,q_j$, then (assuming *it is syntactically correct)*

$$\tilde{f}(\varsigma(v_1, s(q_1, ..., q_j), \mathcal{R}_1), ..., \varsigma(v_k, s(q_1, ..., q_j), \mathcal{R}_k), \tilde{h}_1, ..., \tilde{h}_k)$$

has sensitivity $u_1 + ... + u_k + p_1 + ... + p_j + s(q_1, ..., q_k)$ *and bound* $t(q_1, ..., q_k)$.

e) $x \in Y$, $X \in \mathcal{Y}$, $x \leq y$, $x = y$ *and* $X = Y$ *are permissible formulas with sensitivity polynomials* 0, $|X|$, 0, 0 *and* 0, *respectively.*

f) If $\mathcal{R}$ *is a term of sensitivity* s *and bound* t, *and* H *is a string term of sensitivity* u *and bound* v, *then* $X \in \varsigma(t, |X|, \mathcal{R}))$ *and* $H \in \varsigma(t, v, \mathcal{R})$ *are permissible formulas with sensitivities* s *and* $s + u$ *respectively.*

g) Any other application of a predicate symbols to terms of the appropriate type results in a permissible formula whose sensitivity is the sum of the sensitivities of the given terms.

h) If ϕ *and* θ *are permissible formulas of sensitivity* s *and* t, *then* $\phi \wedge \theta$, $\phi \vee \theta$ *and* $\neg\phi$ *are permissible with sensitivity* $s + t$, $s + t$ *and* s, *respectively.*

Definition 6.0.3 ($\mathcal{L}_{PS}$). $\mathcal{L}_{PS}$ *is the smallest class satisfying*

a) $\mathcal{L}_{PS}$ *includes* $\mathcal{L}_A^3 \cup \{pd, <_2, f_{SE}, \varsigma\}$.

b) For each permissible open formula $\alpha(z, \overline{x}, \overline{X}, \overline{\mathcal{X}})$ *of sensitivity* $s(z)$ *over* $\mathcal{L}_{PS}$ *and number term* t *over* $\mathcal{L}_A^3$, *there is a string function* $F_{\alpha,t,s(t)}$ *of sensitivity* $s(t)$ *and bound* t *with defining axiom*

$$F_{\alpha,t,s(t)}(\overline{x}, \overline{X}, \overline{\mathcal{X}})(z) \leftrightarrow z < t \wedge \alpha(z, \overline{x}, \overline{X}, \overline{\mathcal{X}}) \tag{6.0.1}$$

intended to simulate 2-COMP.

c) For each permissible open formula $\alpha(z, \overline{x}, \overline{X}, \overline{\mathcal{X}})$ *of sensitivity* $s(z)$ *over* $\mathcal{L}_{PS}$ *and number term* t *over* $\mathcal{L}_A^3$ *(free variables among those of* α*), there is a number function* $g_{\alpha,t,s(t)}$

of sensitivity $s(t)$ and bound t with defining axioms

$$g_{\alpha,t,s(t)}(\ldots) \leq t(\ldots) \tag{6.0.2}$$

$$g_{\alpha,t,s(t)}(\ldots) < t(\ldots) \supset \alpha(g_{\alpha,t,s(t)}(\ldots), \ldots) \tag{6.0.3}$$

$$z < g_{\alpha,t,s(t)}(\ldots) \supset \neg\alpha(z, \ldots) \tag{6.0.4}$$

intended to allow elimination of number quantifiers. It follows from these defining axioms that

$$\exists z < t\alpha(z, \ldots) \leftrightarrow g_{\alpha,t,s(t)}(\ldots) < t.$$

A suitable witness for these axioms is $g_{\alpha,t,s(t)}(\ldots) = \min z < t\alpha(z, \ldots)$.

d) For each permissible open formula $\alpha(Z, \ldots)$ of sensitivity $s(|Z|)$ over $\mathcal{L}_{PS}$ and number term t over $\mathcal{L}_A^3$(free variables among those of α), there is a superstring function $\mathcal{F}_{\alpha,t,s(t)}$ of sensitivity $s(t)$ and bound t with defining axiom

$$|Z| \leq t \supset [\mathcal{F}_{\alpha,t,s(t)}(\ldots)(Z) \leftrightarrow \alpha(Z, \ldots)] \tag{6.0.5}$$

intended to simulate 3-COMP.

e) For each permissible open formula $\alpha(Z, \overline{x}, \overline{X}, \overline{\mathcal{X}})$ of sensitivity $s(|Z|)$ over $\mathcal{L}_{PS}$ and number term t over $\mathcal{L}_A^3$ (free variables among those of α, there is a string function $G_{\alpha,t,s(t)}$ of sensitivity $s(t)$ and bound t with defining axioms

$$|G_{\alpha,t,s(t)}(\ldots)| \leq t(\ldots) \tag{6.0.6}$$

$$|G_{\alpha,t,s(t)}(\ldots)| < t(\ldots) \supset \alpha(G_{\alpha,t,s(t)}(\ldots), \ldots) \tag{6.0.7}$$

$$Z <_2 G_{\alpha,t,s(t)}(\ldots) \supset \neg\alpha(Z, \ldots) \tag{6.0.8}$$

intended to allow elimination of string quantifiers. It follows that

$$\exists Z < t\alpha(Z, \ldots) \leftrightarrow |G_{\alpha,t,s(t)}(\ldots)| < t,$$

and a suitable witness is $G_{\alpha,t,s(t)}(\ldots) = \min Z < t\alpha(Z, \ldots)$

f) For each three functions $\mathcal{G}(\ldots), \mathcal{H}(x, \mathcal{Z}, \ldots)$ and $l(x, \ldots)$ of $\mathcal{L}_{PS}$ with sensitivities $s_{\mathcal{G}}, s_{\mathcal{H}}$ and s_l and bounds $t_{\mathcal{G}}, t_{\mathcal{H}}$ and t_l there is a function $\mathcal{F}_{\mathcal{G},\mathcal{H},l}$ of sensitivity $s_{\mathcal{G}}+s_{\mathcal{H}}+s_l$ and bound t_l with defining axioms

$$|Y| \leq l(0, \ldots) \supset [\mathcal{F}_{\mathcal{G},\mathcal{H},l}(0, \ldots)(Y) \leftrightarrow \varsigma(t_{\mathcal{G}}, t_l, \mathcal{G}(\ldots))(Y)] \tag{6.0.9}$$

$$|Y| \leq l(x+1, \ldots) \supset [\mathcal{F}_{\mathcal{G},\mathcal{H},l}(x+1, \ldots)(Y) \leftrightarrow \varsigma(t_{\mathcal{H}}, t_l, \mathcal{H}(x, \mathcal{F}_{\mathcal{G},\mathcal{H},l}(x, \ldots), \ldots))(Y)] \tag{6.0.10}$$

intended to define $\mathcal{F}_{\mathcal{G},\mathcal{H},l}$ by limited recursion from $\mathcal{G}$ and $\mathcal{H}$ with limit l.

Definition 6.0.4. *$\overline{HW_1^0}$ is the universal theory over $\mathcal{L}_{PS}$ consisting of the universal closures of B1-B13, B14' and B14" (open replacements for B14 defining pd), L1, L2, SE' and SE" (the defining axioms of f_{SE}), the defining axiom of ς, and finally all defining axioms 6.0.2–6.0.10 of $\mathcal{L}_{PS}$.*

After the lemma, we show that $\overline{HW_1^0}$ extends HW_1^0.

Lemma 6.0.5. *For every $\Sigma_0^{\mathcal{B}}$ formula ϕ there is an open formula α of $\mathcal{L}_{PS}$ such that $\overline{HW_1^0} \vdash \phi \leftrightarrow \alpha$.*

Proof outline. This follows by structural induction on ϕ, using cases c and e of the definition of $\mathcal{L}_{PS}$. The permissibility of the open formulas constructed is not a factor, as no superstring-valued functions are constructed. □

Theorem 6.0.6. $\overline{HW_1^0} \vdash HW_1^0$

Proof. B14 follows from B14' and B14". That $\Sigma_0^{\mathcal{B}}$-{2,3}COMP are provable follows from the previous lemma and cases b and d of the definition of $\mathcal{L}_{PS}$.

Finally, for any given $\phi \in \Sigma_0^{\mathcal{B}}$, $\mathcal{L}_{PS}$ contains a function witnessing the ϕ-ss-hrc scheme: this function is defined by limited recursion from a function that outputs $\mathcal{X}$ satisfying

$$\forall Y \leq z + 1(\mathcal{X}(Y) \leftrightarrow \phi(Y, \mathcal{X}^{Y/2}))$$

given $\mathcal{X}'$ satisfying

$$\forall Y \leq z(\mathcal{X}'(Y) \leftrightarrow \phi(Y, \mathcal{X}'^{Y/2})).$$

The correctness of this function is proved by open($\mathcal{L}_{PS}$)-IND, which is derived in the standard way (see [14]) in $\overline{HW_1^0}$ from the comprehension (the fact that nonempty sets have a largest element implies a minimization scheme that then implies induction). □

To show that $\overline{HW_1^0}$ is conservative over HW_1^0, we inductively show that every function of $\mathcal{L}_{PS}$ is $\Sigma_1^{\mathcal{B}}$-definable in HW_1^0. In fact, this seems not to be a strong enough induction hypothesis, so we in fact show something stronger: that each function symbol of $\mathcal{L}_{PS}$ is $\Sigma_0^{\mathcal{B}}$-HR-definable, a concept that we now define:

Definition 6.0.7. *Let T be a theory over $\mathcal{L} \supseteq \mathcal{L}_A^3$ and Φ a set of $\mathcal{L}$-formulas. Then a function $\tilde{f}(\overline{x}, \overline{X}, \overline{\mathcal{X}}) \in \mathbb{E}^1 \cup \mathbb{E}^2$ is* **Φ-HR-definable in** *T if there are $\phi_1(Y, \mathcal{Z}, \overline{x}, \overline{X}, \overline{\mathcal{X}})$ and $\phi_2(\tilde{y}, \mathcal{Z}, \overline{x}, \overline{X}, \overline{\mathcal{X}})$ from Φ and term $s(\overline{x}, \overline{X})$ over $\mathcal{L}$ (all free variables displayed) such that*

1. *$T \vdash \forall \overline{x}, \overline{X}, \overline{\mathcal{X}} \exists! \tilde{y} \exists \mathcal{Z} (\phi_1^{hrc}(s(...), \mathcal{Z}, \overline{x}, \overline{X}, \overline{\mathcal{X}}) \wedge \phi_2(\tilde{y}, \mathcal{Z}^{<s(...)}, \overline{x}, \overline{X}, \overline{\mathcal{X}}))$*

2. *$\tilde{f}(\overline{x}, \overline{X}, \overline{\mathcal{X}})$ satisfies the defining formula in the standard model for all values of the parameters.*

Similarly, $\mathcal{F}(\overline{x}, \overline{X}, \overline{\mathcal{X}}) \in \mathbb{E}^3$ is **length $2^{t(\overline{x}, \overline{X})}$ Φ-HR-definable in** *T if there are $\phi_1, \phi_2 \in \Phi$ and s over $\mathcal{L}$ (as above) such that*

1. *$T \vdash \forall \overline{x}, \overline{X}, \overline{\mathcal{X}} \exists \mathcal{Y} \exists \mathcal{Z}$*
   $(\phi_1^{hrc}(s(...), \mathcal{Z}, \overline{x}, \overline{X}, \overline{\mathcal{X}}) \wedge \forall Y \leq t(\overline{x}, \overline{X})(\mathcal{Y}(Y) \leftrightarrow \phi_2(Y, \mathcal{Z}^{<s(...)}, \overline{x}, \overline{X}, \overline{\mathcal{X}})))$

2. *$\mathcal{F}(\overline{x}, \overline{X}, \overline{\mathcal{X}})$ satisfies the defining formula in the standard model for all values of the parameters.*

If $\mathcal{F}$ is length 2^t Φ-HR-definable in T for some true bound t, then we say $\mathcal{F}$ is Φ-HR-definable in T.

Some explanation of the previous definition is in order. A function $\tilde{f}$ that is Φ-HR-definable in a theory T is defined syntactically by a Φ-halfrecursion (computing a superstring) composed with a Φ-definition. In the case that $\Phi = \Sigma_0^{\mathcal{B}}$ and $T = HW_1^0$, this corresponds to defining a superstring with a halfrecursion operation from a PH predicate, and composing with a PH function to produce the final value. These two operations composed in this way can produce every function in FPSPACE$^+$, as the halfrecursion operation can produce the computation of a PSPACE Turing Machine (or an array of computations if a superstring-valued function is to be computed), and then a PH function can extract the output of the machine (or collect the bits of the superstring value of the function from the array of computations).

Now the conservativity of $\overline{HW_1^0}$ over HW_1^0 follows from the following lemma:

Lemma 6.0.8. *The functions of $\mathcal{L}_{PS}$ are all $\Sigma_0^{\mathcal{B}}$-HR-definable in HW_1^0. Furthermore, they are all provably insensitive to their superstring arguments past the claimed sensitivity bounds.*

Proof. This is proved by induction on the definition of the functions, considered in some appropriate enumeration. At each step, the $\Sigma_0^{\mathcal{B}}$-HR-definitions of all relevant function symbols are combined into one $\Sigma_0^{\mathcal{B}}$-HR-definition of the new function symbol.

Consider for example a function symbol $\mathcal{F}$ defined from a permissible formula α and $\mathcal{L}_A^3$-term t using case d) of the definition of $\mathcal{L}_{PS}$. By the induction hypothesis, each function symbol in α is $\Sigma_0^{\mathcal{B}}$-HR-definable in HW_1^0. Now all the formulas ϕ_1 and ϕ_2 from the $\Sigma_1^{\mathcal{B}}$-HR-definitions of these functions symbols can be combined into one $\Sigma_0^{\mathcal{B}}$ formula ϕ, such that ϕ^{hrc} asserts the existence of one large superstring computing the value of $\mathcal{F}$. This large superstring contains subcomputations for each occurrence of a function symbol in α, arranged in some suitable order of evaluation; each such subcomputation is followed by another phase extracting the output of the corresponding function symbol occurrence from the computation. Bounds on the lengths of these computations and outputs are all known in advance, so the ϕ_1 and ϕ_2 formulas can be amended to reference their respective parts of this big superstring. Finally, a $\Sigma_1^{\mathcal{B}}$-formula extracts the result from the end of this large computation. Since α is permissible, at every step the computation is provably well defined (i.e., provably depends only on bits of superstring outputs actually fixed by defining axioms of the appropriate function symbols). Thus by induction on the structure of α, $\mathcal{F}$ is uniquely defined to the given bound.

Case b) is similar. Cases c) and e) use the function symbols from cases b) and d) respectively to construct a table of values of α on all inputs up to the given bound and then extract the minimum value satisfying α. The defining axioms are then proved in HW_1^0 directly from the definition of this table.

Finally, a function symbol $\mathcal{F}$ defined using case f) is computed by limited recursion. Again, one large superstring records, one after the other, the computations of each step of this recursion (and there are polynomially many). The $\Sigma_0^{\mathcal{B}}$-HR-definition of $\mathcal{F}$ asserts that this superstring exists and is defined appropriately, and the value of $\mathcal{F}$ is extracted by a $\Sigma_0^{\mathcal{B}}$-bit-definition. $\square$

Chapter 7

Witnessing Theorems

7.1 Sequent Calculus Formulations

In this chapter we prove some Buss-style witnessing theorems.

We begin by introducing some equivalent sequent formulations of several theories. LK^3 is like the system LK, but with the addition of the following quantifier introduction rules:

$$\forall : \textbf{left} \quad \frac{\phi(\tilde{Y}), \Gamma \longrightarrow \Delta}{\forall \tilde{X} \phi(\tilde{X}), \Gamma \longrightarrow \Delta} \qquad \text{and} \qquad \exists : \textbf{right} \quad \frac{\Gamma \longrightarrow \Delta, \phi(\tilde{Y})}{\Gamma \longrightarrow \Delta, \exists \tilde{X} \phi(\tilde{X})}$$

and

$$\exists : \textbf{left} \quad \frac{\phi(\tilde{Y}), \Gamma \longrightarrow \Delta}{\exists \tilde{X} \phi(\tilde{X}), \Gamma \longrightarrow \Delta} \qquad \text{and} \qquad \forall : \textbf{right} \quad \frac{\Gamma \longrightarrow \Delta, \phi(\tilde{Y})}{\Gamma \longrightarrow \Delta, \forall \tilde{X} \phi(\tilde{X})}$$

where $\tilde{X}$ and $\tilde{Y}$ are either both second- or both third-order variables, and in the latter two rules $\tilde{Y}$ may not occur in the conclusion of the inference. Formally, LK^3 also adopts the usual conventions concerning free and bound variables, as in [8].

The system $\mathrm{LK}^3 - W_1^i$ additionally includes the $\forall^2\Sigma_i^{\mathcal{B}}$-IND rule:

$$\frac{\Gamma, \phi(b) \longrightarrow \phi(b+1), \Delta}{\Gamma, \phi(0) \longrightarrow \phi(t), \Delta},$$

where b appears only as indicated and $\phi \in \forall^2\Sigma_i^{\mathcal{B}}$. As initial sequents we allow all substitution instances of the axioms (other than induction) of W_1^i. Note that all rules of $\mathrm{LK}^3 - W_1^i$

are valid in W_1^i, and furthermore, $\mathrm{LK}^3 - W_1^i$ proves the induction and comprehension schemes of W_1^i.

The system $\mathrm{LK}^3 - \widehat{W_1^i}$ is as above, but with the $\Sigma_i^{\mathcal{B}}$-IND rule instead.

The system $\mathrm{LK}^3 - TW_1^i$ is as above, but with the $\Sigma_i^{\mathcal{B}}$-SIND rule instead. This rule is:

$$\frac{\Gamma, \phi(X), S(X,Y) \longrightarrow \phi(Y), \Delta}{\Gamma, |X| = 0, \phi(X) \longrightarrow \phi(Z), \Delta},$$

where X, Y appear only as indicated and $\phi \in \Sigma_i^{\mathcal{B}}$.

The system $\mathrm{LK}^3 - HW_1^0$ is LK^3 plus, as initial sequents, all substitution instances of axioms of HW_1^0.

The standard definition of an anchored cut in LK^3 is extended in the usual way for any of the above systems by allowing cuts on the descendants of principal formulas of the appropriate induction rule, in addition to cuts on descendants of formulas in nonlogical axioms. The anchored completeness theorem for LK^3 can be extended to these systems in the usual way to cope with the induction rules, as detailed in [56].

7.2 A Witnessing Theorem for W_1^i

In this section we prove a Buss-style witnessing theorem showing that for $i \geq 1$, every $\Sigma_i^{\mathcal{B}}$-definable function of W_1^i is in $(\mathrm{FPSPACE}^{(\Sigma_{i-1}^{exp})^\diamond})^+$. The witnessing theorem we wish to prove is:

Theorem 7.2.1. *Every $\Sigma_i^{\mathcal{B}}$-definable function of W_1^i is in $(FPSPACE^{(\Sigma_{i-1}^{exp})^\diamond})^+$.*

For $i = 1$, the argument is somewhat easier than for $i > 1$ and a witnessing lemma such as Lemma 7.2.5 below is used directly to yield a $\mathrm{FPSPACE}^+$ witnessing function; this is our result from [55]. Although we focus now instead on the more difficult case of $i > 1$, we nevertheless include the special case $i = 1$ in the statements of Theorem 7.2.2 and Lemma 7.2.5, and comment on how the argument differs.

The proof of Theorem 7.2.1 will require several steps. The first step is to define a slightly stronger complexity class: $(\mathrm{FEXP}^{(\Sigma_{i-1}^{exp})^{\diamond}}[\text{wit,poly}])^{+}$ is like $(\mathrm{FEXP}^{(\Sigma_{i-1}^{exp})^{\diamond}})^{+}$, but with the following two important differences: one, the exponential-time machine is restricted to only polynomially many queries to its third-order oracle; and two, if the answer to an oracle query is "yes", then a witness to the $(\Sigma_{i-1}^{\mathcal{B}})^{\diamond}$ query is returned on a special read-only tape. The computational objects in $\mathrm{FEXP}^{(\Sigma_{i-1}^{exp})^{\diamond}}[\text{wit,poly}])^{+}$ are now multi-functions as there could be many possible witnesses to a given oracle query, which could of course affect the progress of the computation.

The second step is to prove the following witnessing theorem. This next theorem differs from Theorem 7.2.1 in that there is one less assumption: that the $\tilde{y}$ is provably unique. Consequently, the conclusion is weaker.

Theorem 7.2.2. *Let $i > 0$ and Suppose $W_1^i \vdash \exists \tilde{y}\phi(\overline{\tilde{x}}, \tilde{y})$, for $\phi(\overline{\tilde{x}}, \tilde{y}) \in \Sigma_i^{\mathcal{B}}$ with all free variables displayed, and $\overline{\tilde{x}}, \tilde{y}$ of any sort. Then there exists a function $\tilde{f}$ in the class $(FEXP^{(\Sigma_{i-1}^{exp})^{\diamond}}[wit,poly])^{+}$ such that for all values of $\overline{\tilde{x}}$, $\phi(\overline{\tilde{x}}, \tilde{f}(\overline{\tilde{x}}))$ is true in the standard model for every possible output of $\tilde{f}$. If $i = 1$, this function is in $FPSPACE^{+}$.*

The final step in the proof of Theorem 7.2.1 is as follows: Assume that a function $\tilde{f}$ is $\Sigma_i^{\mathcal{B}}$-definable in W_1^i. If $i = 1$ then the special case of Theorem 7.2.2 gives a $\mathrm{FPSPACE}^{+}$ witnessing function and so there is nothing more to prove. Otherwise, applying Theorem 7.2.2, we know that $\tilde{f} \in (\mathrm{FEXP}^{(\Sigma_{i-1}^{exp})^{\diamond}}[\text{wit,poly}])^{+}$, computed by exponential-time Turing machine M (with an oracle for $(\Sigma_{i-1}^{\mathcal{B}})^{\diamond}$).

Using the additional assumption that the output of $\tilde{f}$ is unique (boundedly so in the case of a superstring-valued function), we now show that in fact $\tilde{f} \in (\mathrm{FPSPACE}^{(\Sigma_{i-1}^{exp})^{\diamond}})^{+}$ by describing a PSPACE Turing machine M' that computes $\tilde{f}$. (Note that this only works for $i > 1$!) First, M' on input $\overline{\tilde{x}}$ determines the lexicographically maximal sequence of yes oracle answers of M by binary search, asking (non-witness) queries of the form "does there exist a sequence of oracle answers, a computation of M and witnesses to the yes answers that are all consistent?". Following that, M' computes each bit of the output of

M' in turn by asking if there exist a computation of M and witnesses for the yes answers to the relevant queries such that the desired bit of the output of M is 1.

Now all that remains is to prove Theorem 7.2.2. For this we shall need several definitions:

Definition 7.2.3. *Let $\psi \equiv \forall X \leq t \exists \mathcal{X} \phi(X, \mathcal{X}) \in \forall^2 \Sigma_i^{\mathcal{B}}$, with other free variables not shown. Consider an assignment to the free variables of ψ. Then the superstring $\mathcal{A}$* **satisfies** *ψ (with respect to the assignment to the free variables of ψ) iff for every string A of no more than t bits, $\phi(A, \mathcal{A}^{[A]})$ is true in the standard model.*

Definition 7.2.4. *Let S be the sequent $\Gamma \longrightarrow \Delta$ such that $\Gamma \bigcup \Delta \subset \forall^2 \Sigma_i^{\mathcal{B}}$, i.e.*

$$\Gamma = \{\forall A_j \leq s_j \exists \mathcal{A}_j \gamma_j(A_j, \mathcal{A}_j, \bar{\bar{b}})\}$$

and

$$\Delta = \{\forall C_j \leq t_j \exists \mathcal{C}_j \delta_j(C_j, \mathcal{C}_j, \bar{\bar{b}})\},$$

with $\{\gamma_j\} \bigcup \{\delta_j\} \subset \Sigma_{i-1}^{\mathcal{B}}$, and although we write for simplicity the initial string and third-order quantifiers for each formula, in fact for some of the formulas either the initial string quantifier or both initial quantifiers may be absent.

*Then i-***Witnessing Functions (WFs)** *for S are superstring-valued functions: For each formula from Δ*

$$\forall C_j \leq t_j \exists \mathcal{C}_j \delta_j(C_j, \mathcal{C}_j, \bar{\bar{b}})$$

which is not $\Sigma_{i-1}^{\mathcal{B}}$ (and may or may not have the leading string quantifier as pictured), the WF $\mathcal{F}_j$ is a function with arguments $\bar{\bar{b}}$ (for the free variables of the sequent) and $\{\mathcal{A}_k\}$ (for the superstrings satisfying the formulas in the antecedent). The $\mathcal{F}_j$ must have the property that for any assignment to the free variables $\bar{\bar{b}}$ of S and superstrings $\{\mathcal{A}_k\}$, if each formula γ_k is satisfied by the corresponding $\mathcal{A}_k$, then some δ_j is satisfied by the output of $\mathcal{F}_j(\overline{\mathcal{A}}, \bar{\bar{b}})$.

Observe that the above definition of a witnessing function for a sequent is simplified by the following factors: First, the relatively restricted class of formulas in the sequents to be witnessed; and second, the availability of third-order functions. Without a definition of third-order computation, these functions could be tricky to define: see [55] for one method.

Now the theorem will follow from the following lemma:

Lemma 7.2.5. *Suppose $LK^3 - W_1^i \vdash \Gamma \longrightarrow \Delta$, where $\Gamma \bigcup \Delta \subset \forall^2\Sigma_i^{\mathcal{B}}$. Then there exist i-witnessing functions (actually, multi-functions) from $(FEXP^{(\Sigma_{i-1}^{exp})^\diamond}[wit,poly])^+$ for $\Gamma \longrightarrow \Delta$. If $i = 1$, the witnessing functions are in $FPSPACE^+$.*

Proof of Theorem 7.2.2 from Lemma 7.2.5. Suppose $W_1^i \vdash \exists\tilde{y}\phi(\overline{\tilde{x}}, \tilde{y})$, for $\phi(\overline{\tilde{x}}, \tilde{y}) \in \Sigma_i^{\mathcal{B}}$ with all free variables displayed. We have two cases:

If $\tilde{y}$ is a number or string variable, then by Parikh's theorem, $W_1^i \vdash \exists\tilde{y} \leq t(\overline{\tilde{x}})\phi(\overline{\tilde{x}}, \tilde{y})$, for some term t. By Corollary 4.5.4, $W_1^i \vdash \phi(\overline{\tilde{x}}, \tilde{y}) \leftrightarrow \exists\mathcal{Z}\psi(\overline{\tilde{x}}, \tilde{y}, \mathcal{Z})$, for some $\psi \in \Sigma_{i-1}^{\mathcal{B}}$. Also,

$$W_1^i \vdash \exists\tilde{y} \leq t(\overline{\tilde{x}})\exists\mathcal{Z}\psi(\overline{\tilde{x}}, \tilde{y}, \mathcal{Z}) \leftrightarrow \exists\mathcal{Z}\exists\tilde{y} \leq t(\overline{\tilde{x}})\psi(\overline{\tilde{x}}, \tilde{y}, \mathcal{Z}).$$

Applying the lemma to the sequent

$$\longrightarrow \exists\mathcal{Z}\exists\tilde{y} \leq t(\overline{\tilde{x}})\psi(\overline{\tilde{x}}, \tilde{y}, \mathcal{Z}),$$

we obtain a superstring-valued multi-function $\mathcal{F}$ (of $\overline{\tilde{x}}$) from $(\mathrm{FEXP}^{(\Sigma_{i-1}^{exp})^\diamond}[\mathrm{wit,poly}])^+$ satisfying that sequent, and so for particular $\overline{\tilde{x}}$ the number or string $\tilde{y}$ can be obtained by evaluating $\psi(\overline{\tilde{x}}, \tilde{y}, \mathcal{F}(\overline{\tilde{x}}))$ on each number (or string of length) $\leq t(\overline{\tilde{x}})$ in turn. Each such evaluation is a $(\Sigma_{i-1}^{exp})^\diamond$ predicate composed with a $(\mathrm{FEXP}^{(\Sigma_{i-1}^{exp})^\diamond}[\mathrm{wit,poly}])^+$ function (which need only be computed once), and polynomial space suffices to keep track of the iterations; the total number of oracle queries required does not increase. The entire computation is thus in $(\mathrm{FEXP}^{(\Sigma_{i-1}^{exp})^\diamond}[\mathrm{wit,poly}])^+$. It is easy to see that the computed value $\tilde{y}$ satisfies $\phi(\overline{\tilde{x}}, \tilde{y})$ (for the same fixed $\overline{\tilde{x}}$).

Now, if $\tilde{y}$ is $\mathcal{Y}$, a superstring variable, then as above but skipping the application of Parikh's Theorem, $W_1^i \vdash \exists \mathcal{Y} \phi(\overline{\tilde{x}}, \mathcal{Y}) \leftrightarrow \exists \mathcal{Y} \exists \mathcal{Z} \psi(\overline{\tilde{x}}, \mathcal{Y}, \mathcal{Z}) \leftrightarrow \exists \mathcal{Z} \psi(\overline{\tilde{x}}, \mathcal{Z}^{[0]}, \mathcal{Z}^{[1]})$, for some $\psi \in \Sigma_{i-1}^{\mathcal{B}}$. Applying the lemma we again obtain a multi-function $\mathcal{F}$ from $(\mathrm{FEXP}^{(\Sigma_{i-1}^{exp})^\diamond}[\mathrm{wit,poly}])^+$ satisfying the sequent, and now after computing $\mathcal{F}$ and obtaining one possible value of $\mathcal{F}(\overline{\tilde{x}})$, we simply output $\mathcal{F}(\overline{\tilde{x}})^{[0]}$ to the appropriate length (determined by the bounding terms in ψ).

For the special case $i = 1$ the proof is similar, but when evaluating $\psi(\overline{\tilde{x}}, \tilde{y}, \mathcal{F}(\overline{\tilde{x}}))$ on possible values of $\tilde{y}$, the bits of $\mathcal{F}(\overline{\tilde{x}})$ are computed on demand by the FPSPACE$^+$ WF, as they cannot be computed once and stored. $\square$

All that remains is to prove the lemma:

Proof of Lemma 7.2.5. Suppose $LK^3 - W_1^i \vdash \Gamma \longrightarrow \Delta$, where $\Gamma \bigcup \Delta \subset \forall^2\Sigma_i^{\mathcal{B}}$, and consider an anchored proof π of this sequent. Since both the endsequent of π and every nonlogical axiom of $LK^3 - W_i^1$ is $\forall^2\Sigma_i^{\mathcal{B}}$, and since the induction rule is limited to this same class of formulas, every formula in π is $\forall^2\Sigma_i^{\mathcal{B}}$.

We now show by induction on the number of sequents in π that WFs from the class $(\mathrm{FEXP}^{(\Sigma_{i-1}^{exp})^\diamond}[\mathrm{wit,poly}])^+$ exist for $\Gamma \longrightarrow \Delta$. For $i = 1$ the only differences occur in the induction rule, and our comments on this special case are found there; other cases apply to FPSPACE$^+$ WFs directly.

Base Case: The base case is that $\Gamma \longrightarrow \Delta$ is either an initial sequent of LK^3 or an instance of an axiom. The only such sequents requiring WFs are those with a third-order quantifier in the succedent, namely an instance

$$\longrightarrow (\exists \mathcal{Y})(\forall Z \leq s(\overline{\tilde{b}}))[\phi(\overline{\tilde{b}}Z) \leftrightarrow \mathcal{Y}(Z)]$$

of $\Sigma_0^{\mathcal{B}}$-3COMP, where $\phi \in \Sigma_0^{\mathcal{B}}$, subject to the restriction that $\mathcal{Y}$ does not occur free in ϕ. The only WF required for this sequent is computed by limited 3-comprehension on the predicate

$$|Z| \leq s(\overline{\tilde{b}}) \wedge \phi(\overline{\tilde{b}}Z),$$

which is in some level of the polynomial-time hierarchy, and thus certainly in the class $(\mathrm{FEXP}^{(\Sigma^{exp}_{i-1})^{\diamond}}[\mathrm{wit,poly}])^{+}$

Induction Step: The induction step has several cases depending on which rule has been used to derive $\Gamma \longrightarrow \Delta$.

1. Weakening:

   The WFs from the hypothesis are modified to take any extra arguments the new formula introduces (free variables or an existential third-order quantifier in the antecedent) and to ignore them. If the formula is added to the succedent and contains a third-order quantifier, a function taking the appropriate arguments and returning an empty superstring is added as the new WF for the conclusion.

2. Contraction:

   If the contraction occurs in the succedent on a formula ϕ with a third-order quantifier, then one less WF is required for the conclusion. Construct a new WF for ϕ which evaluates ϕ on each of the two original WFs in turn (each evaluation is computable in $(\mathrm{FEXP}^{(\Sigma^{exp}_{i-1})^{\diamond}}[\mathrm{wit,poly}])^{+}$) and then behaves like whichever satisfies ϕ, if any. This computation requires only the sum of the time and oracle queries used by the two original WFs.

   If the contraction occurs in the antecedent on a formula ϕ with a third-order quantifier, then all original WFs must be modified to accept one less superstring argument. Each is modified to compute the original WF but now passing the superstring argument from ϕ twice.

3. Exchange, introduction of $\neg$, $\vee$ on the right and $\wedge$ on the left:

   These rules can neither introduce nor eliminate free variables. No third-order quantifiers are added or removed, and no formula with a third-order quantifier is changed, so the WFs from the hypothesis are used without modification for the conclusion.

4. Introduction of $\vee$ on the left and $\wedge$ on the right:

   These inferences have two hypotheses, and the principal formula is $\Sigma^{\mathcal{B}}_{i-1}$ and so needs no WF. Any side formula which is not $\Sigma^{\mathcal{B}}_{i-1}$ will have a WF for each hypothesis. As in the case of contraction, the WF for such a formula in the conclusion evaluates the formula using each WF from the hypotheses, and then computes whichever satisfies it, if any.

5. First- or second-order $\forall$: **left** and $\exists$: **right**:

   The conclusion of such an inference may have less free variables than the hypothesis. Taking for example an $\exists$: **right** inference with principal formula $\exists X\phi(X)$ with the corresponding formula in the hypothesis being $\phi(B)$ and B not free in the conclusion, all WFs for the hypothesis will have B as an argument. If this argument is fixed to the empty string, the resulting set of WFs will suffice for the conclusion of the inference (unless $\phi \notin \Sigma^{\mathcal{B}}_{i-1}$, addressed below). $\forall$: **left** is similar and in the first-order cases one analogously substitutes 0 for eliminated variables.

   If $\phi \notin \Sigma^{\mathcal{B}}_{i-1}$ then the principal formula of the inference is $\forall A_j \leq s_j \exists \mathcal{A}_j \gamma_j(A_j, \mathcal{A}_j, \bar{\bar{b}})$ and occurs in the antecedent. In addition to the procedure above (substituting the empty string for the eliminated free string variable), the WFs must be modified so that any query to $\mathcal{A}_j(X)$ becomes $\mathcal{A}_j^{[\lambda]}(X)$, adding the empty string as an additional argument, since in the conclusion this superstring argument to the WFs codes an array.

6. First- or second-order $\forall$: **right** and $\exists$: **left**:

   As in the previous case free variables are eliminated by such inferences. However, it is not sufficient to substitute a dummy value for them as above since such a value would not witness the new quantifier properly. For example, if the new quantifier is universal on the right and the principal formula is false under some assignment, the WFs (from the hypothesis) for the remaining formulas expect a value falsifying the principal formula. This value is found by a query to the $\Sigma^{\mathcal{B}}_{i-1}$ witness oracle, as the formula to be falsified or satisfied is $\Sigma^{\mathcal{B}}_{i-1}$, following which the WFs for the conclusion evaluate the WFs from the hypothesis.

If the principal formula is not $\Sigma^{\mathcal{B}}_{i-1}$, then it is $\forall C_j \leq t_j \exists \mathcal{C}_j \delta_j(C_j, \mathcal{C}_j, \bar{\bar{b}})$ and is in the succedent. In this one special case the WF for δ_j has one less argument in the conclusion, due to the string quantifier preceding the third-order quantifier. The WF for δ_j alone is not modified as above, but instead computes the previous WF over all possible values of C_j and outputs an array of all the values obtained.

7. Third-order $\exists$: **left**:

   The principal formula is $\exists \mathcal{A}_j \gamma_j(A_j, \mathcal{A}_j, \overline{\mathcal{B}}, \overline{B}, \bar{b})$. All WFs from the antecedent are modified to accept superstring argument $\mathcal{A}_j$ instead of the free third-order variable eliminated by the quantifier introduction.

8. Third-order $\exists$: **right**:

   If the eigenvariable $\mathcal{B}$ occurs in the lower sequent, then the WF for the principal formula is defined by

$$f(\bar{\bar{b}}, \overline{\mathcal{A}}, Z) \leftrightarrow \mathcal{B}(Z)$$

   If not, analogously to the lower-order cases of this rule, the new quantifier is witnessed by any value and thus the WF for the new quantifier may ignore its arguments and always return false. Furthermore, a constant-false predicate is supplied in the place of the eliminated variable as an argument to the other WFs from the hypothesis.

9. The **cut** rule:

   The inference is

$$\frac{\Gamma \longrightarrow \phi, \Delta \qquad \Gamma, \phi \longrightarrow \Delta}{\Gamma \longrightarrow \Delta}.$$

A WF for the conclusion proceeds in two phases: First, it evaluates its formula using the WF from the left hypothesis, and if the output of that WF satisfies the formula, it is returned as output. Otherwise, it emulates the WF from the right hypothesis, and uses the WF for ϕ from the left hypothesis to supply a value for the superstring argument. The whole procedure uses at most the sum of the time and oracle query requirements of the two WFs from the hypotheses, plus a constant number of oracle queries to evaluate formulas.

If any free variables are eliminated, then as before a dummy argument of the correct type is supplied to the WFs.

10. $\forall^2\Sigma_i^{\mathcal{B}}$-IND:

The inference is:

$$\frac{\Gamma, \phi(b) \longrightarrow \phi(b+1), \Delta}{\Gamma, \phi(0) \longrightarrow \phi(t), \Delta}.$$

The WFs for the conclusion will iterate the construction from the previous case, as the current instance of the induction rule could be simulated by t instances of the cut rule, along with some weakenings.

More precisely, let $\mathcal{F}_\phi$ be the WF for the instance of ϕ in the succedent of the hypothesis. Let ψ be any formula in the succedent of the hypothesis (including ϕ) and $\mathcal{F}_\psi$ its WF. We construct a WF $\mathcal{F}'_\psi$ for ψ in the conclusion in stages:

$\mathcal{F}_\psi(0) = \mathcal{F}_\psi$ (other arguments suppressed).

$\mathcal{F}_\psi(k) =$ (if $\psi(\mathcal{F}_\psi(k-1))$ then $\mathcal{F}_\psi(k-1)$ else $\mathcal{F}_\psi(k-1, \mathcal{F}_\phi)$).

$\mathcal{F}_\psi(1)$ checks if $\mathcal{F}_\psi$ satisfies ψ and if so, simulates $\mathcal{F}_\psi$. If not, $\mathcal{F}_\psi(1)$ computes $\mathcal{F}_\psi(\mathcal{F}_\phi)$, that is to say, uses $\mathcal{F}_\phi$ to provide the superstring argument corresponding to ϕ.

$\mathcal{F}_\psi(k)$ checks if $\mathcal{F}_\psi(k)$ satisfies ψ and if so, simulates $\mathcal{F}_\psi(k-1)$. If not, $\mathcal{F}_\psi(k)$ computes $\mathcal{F}_\psi(k-1, \mathcal{F}_\phi)$.

$\mathcal{F}'_\psi$, then, evaluates t and computes $\mathcal{F}_\psi(t)$. Computing $\mathcal{F}_\psi(t)$ requires a factor of t more time and oracle queries than required to compute $\mathcal{F}_\phi$ (to compose the functions and evaluate ψ at each step) plus the requirements of $\mathcal{F}_\psi$, and so only increases the time and queries of WFs by a polynomial factor.

For the special case $i = 1$, the WFs for the conclusion are defined as above, but we must argue that they are computable in FPSPACE$^+$. A WF for the conclusion consists, essentially, of the composition of a polynomial number of FPSPACE$^+$ functions; it can be computed by a recursive procedure of this depth, where each instance of the recursion uses polynomial space.

□

7.3 Witnessing for TW_1^i and TTW_1^0

For TW_1^i and TTW_1^0, the result we prove is obtained directly as for Theorem 7.2.2, without the extra work of Theorem 7.2.1. First for TW_1^i:

Theorem 7.3.1. *Suppose $TW_1^i \vdash \exists\tilde{y}\phi(\overline{\tilde{x}}, \tilde{y})$, for $\phi(\overline{\tilde{x}}, \tilde{y}) \in \Sigma_i^{\mathcal{B}}$ with all free variables displayed, and $\overline{\tilde{x}}, \tilde{y}$ of any sort. Then there exists a function $\tilde{f} \in (FEXP^{(\Sigma_{i-1}^{exp})^\diamond})^+$ such that for all values of $\overline{\tilde{x}}$, $\phi(\overline{\tilde{x}}, \tilde{f}(\overline{\tilde{x}}))$ is true in the standard model.*

Proof Outline. This theorem is proved analogously to Theorem 7.2.2, which is to say with a witnessing lemma. All cases of the lemma except for induction and introduction of a string quantifier are the same (or easier, as only $\Sigma_i^{\mathcal{B}}$-formulas are present) and apply to WFs from $(\mathrm{FEXP}^{(\Sigma_{i-1}^{exp})^\diamond})^+$: definition by cases, evaluating $\Sigma_{i-1}^{\mathcal{B}}$-formulas, etc. For the introduction of a universal string quantifier on the right, or an existential one on the left, we can no longer use a witness oracle query to find the correct string. In this case the new WF will simply evaluate the formula on all strings up to the bound, increasing the time requirements by an exponential factor.

In the case of the stronger induction, we must now iterate the WFs from the hypothesis an exponential number of times (i.e., up to a count given by a string in binary). The function performing this iteration again uses an exponential factor more time than the original functions. $\square$

And now for TTW_1^0:

Theorem 7.3.2. *Suppose $TTW_1^0 \vdash \exists\tilde{y}\phi(\overline{\tilde{x}}, \tilde{y})$, for $\phi(\overline{\tilde{x}}, \tilde{y}) \in \Sigma_1^{\mathcal{B}}$ with all free variables displayed, and $\overline{\tilde{x}}, \tilde{y}$ of any sort. Then there exists a function $\tilde{f}$ $\in$FEXP$^+$ such that for all values of $\overline{\tilde{x}}$, $\phi(\overline{\tilde{x}}, \tilde{f}(\overline{\tilde{x}}))$ is true in the standard model.*

Proof Outline. As for TW_1^1 we use a witnessing lemma, and all cases are the same this time except for induction. Rather than an induction rule, now the induction axioms are $\Sigma_1^{\mathcal{B}}$ formulas and so we simply allow (all substitution instances of) them as initial sequents. An instance of the induction axiom is witnessed by a binary search for the superstring falsifying the induction step; this binary search has an exponential number of steps, each time fixing one bit of the superstring, and at each step a $\Sigma_0^{\mathcal{B}}$ formula is evaluated, so the entire procedure is computable in exponential time. $\square$

7.4 Witnessing for HW_1^0

The following witnessing theorem for HW_1^0 is a corollary of that for W_1^1 since $W_1^1 \vdash HW_1^0$. Nevertheless, we outline a direct proof as it illustrates somewhat the computational nature of the $\Sigma_0^{\mathcal{B}}$-superstring-halfrecursion scheme.

Theorem 7.4.1. *Suppose $HW_1^0 \vdash \exists\tilde{y}\phi(\overline{\tilde{x}}, \tilde{y})$, for $\phi(\overline{\tilde{x}}, \tilde{y}) \in \Sigma_1^{\mathcal{B}}$ with all free variables displayed, and $\overline{\tilde{x}}, \tilde{y}$ of any sort. Then there exists a function $\tilde{f}$ $\in$FPSPACE$^+$ such that for all values of $\overline{\tilde{x}}$, $\phi(\overline{\tilde{x}}, \tilde{f}(\overline{\tilde{x}}))$ is true in the standard model.*

Proof. Analogously to the previous theorems: All cases of the previous witnessing lemma are the similar for the present one, except of course the induction rule is now much more restricted. We need one additional case for the $\Sigma_0^{\mathcal{B}}$-superstring-halfrecursion scheme. A witnessing function for the superstring quantifier $\exists\mathcal{X}$ on an instance of this scheme computes a bit of $\mathcal{X}$ by evaluating the $\Sigma_0^{\mathcal{B}}$ formula ϕ from the scheme, and recursively computing the bits of $\mathcal{X}$ required by ϕ. Modulo the recursive calls, this computation is clearly in the PH. Now, the depth of the recursion is only polynomial, as each recursive call halves the relevant number of bits of $\mathcal{X}$. The witnessing function then iterates this process for each bit of $\mathcal{X}$. This entire procedure is thus computable in polynomial space. $\square$

Chapter 8

Propositional Translations

In this chapter we discuss propositional translations of third-order theories. To begin with, we give our translation from [55] of Σ^B_∞-theorems of W^1_1 into BPLK, as an analogue of the translation of Σ^b_∞-theorems of U^1_2 into G, from [42]. Following that, we present a much more general translation of theorems of W^i_1 into families of proofs in a quantified Boolean program proof system, QBP_i; this more general translation uses quantification over Boolean functions, and is in some ways reminiscent of the Protothetic of Stanisław Leśniewski [43].

8.1 Σ^B_∞-Theorems of W^1_1

Although our aim in this section is to translate the Σ^B_∞-theorems of W^1_1 into BPLK, we must in fact define a slightly stronger translation. This is because the language of Boolean programs seems to be inherently unable to express directly the translation of a formula with third-order variables. A slight generalization of Boolean programs, however, does allow free third-order variables to be translated: allowing some function symbols to remain "free", or not defined by the Boolean program, and therefore to represent an arbitrary Boolean function of a particular arity. Although this translation would still seem not to be general enough for a W^1_1 proof, in which sequents in general contain

$\Sigma_1^{\mathcal{B}}$-formulas, an analogue of the argument of [42] applies in our case also; by means of a witnessing argument, we show that a FPSPACE$^+$ function, in the form of a Boolean program function symbol, can provably witness the existential third-order quantifiers in the succedent of each sequent in the proof. This function symbol takes as "input" (is defined from) the free function symbols witnessing the existential quantifiers in the antecedent, as well as free variables. Although analogous to the translation of U_2^1 into the language of G, wherein the propositional language is again insufficiently expressive, in our case the use of free function symbols allows a significant simplification of the proof; in the case of G this is handled by the cumbersome method of asserting that for any formula (to be substituted in place of the inexpressible quantifier on the left), G can prove that some PSPACE function computes a witness for the inexpressible quantifier on the right.

Therefore we define a translation of $\Sigma_0^{\mathcal{B}}$ formulas in the language $\mathcal{L}_A^3$ of W_1^1 (i.e. possibly with free third-order variables, but no third-order quantifiers) into families of propositional sequents in the language of Boolean programs. We then prove a witnessing-style lemma implying that if $\phi(A) \in \Sigma_\infty^B$ and if $W_1^1 \vdash \phi(A)$ then BPLK has short proofs of the translations of ϕ. Our aim is the following theorem:

Theorem 8.1.1. *If $\phi(A) \in \Sigma_\infty^B$ and if $W_1^1 \vdash \phi(A)$ then BPLK has polynomial-sized proofs of the translations $||\phi||$; furthermore, these proofs are definable in S_2^1 and V^1 (or any theory defining polytime functions).*

The theorem will follow from the lemma below. The definability of the proofs follows from the fact that they are easily constructed in polynomial time. It is important to observe that no "free" function symbols will actually occur in the final proof (of the translation of the Σ_∞^B-theorem).

First, we can extend the definitions of a Boolean Program and of a BPLK proof as follows:

Definition 8.1.2. *A* **Boolean semiprogram** *is like a Boolean program, except we allow that some function symbols used in the program be undefined ("free").*

Definition 8.1.3. *A* **BPLK-sequence** *is the same as a BPLK proof except that the requirement that all function symbols occurring in the sequence be defined by the accompanying Boolean program is dropped. Furthermore, the accompanying Boolean program is instead a Boolean semiprogram. Any undefined function symbol appearing in the sequence or the semiprogram is called "free".*

The translation we shall use is below. If P_1 and P_2 are Boolean semiprograms, then $P_1 \diamond P_2$ denotes the result of concatenating P_1 and P_2 and removing duplicate definitions. If the two semiprograms had conflicting definitions for a function symbol, this operations would not be well defined; however, the function symbols used in the translations are given particular names according to their definitions, and so it will always be the case that any two definitions of a function symbol are identical.

Definition 8.1.4. *Let $\phi(\mathcal{A}_1, ..., \mathcal{A}_j, A_1, ..., A_k)$ be $\Sigma_0^{\mathcal{B}}$ in the language $\mathcal{L}_A^3$. For parameters $m_1, ..., m_k, b_1, ..., b_j \in \mathbb{N}$ we construct a Boolean semiprogram $P_{\phi,b_1,...,b_j}^{m_1,...,m_k}$ and a formula $||\phi||_{b_1,...,b_j}^{m_1,...,m_k}$ in the language of Boolean programs, with the atoms $\overline{\overline{p}} = (\overline{p_i}, i = 1, ..., k)$, where each $\overline{p_i} = (p_{i,0}, ..., p_{i,m_k})$. The m_i are intended to represent the lengths of the strings (i.e., least upper bounds of the finite sets) in the instance to be translated, while the b_i represent bounds on the lengths of strings contained in the relevant superstring. For some values of the m_i and b_i, the translation will be undefined (when the superstring bounds are too small).*

By induction on the structure of ϕ:

- *If ϕ is the atomic formula $s = t$ then s and t are first-order terms with no free first-order variables. Third-order variables do not appear in first-order terms so all variable occurrences in s and t are of the form $|A_i|$ for some second-order variable A_i. Then using the value m_i for this subterm the terms s and t can be evaluated to*

$\underline{s}$ *and* $\underline{t}$. *We define* $||s=t||^{m_1,...,m_k} := 1$ *if* $\underline{s}=\underline{t}$ *and* $||s=t||^{m_1,...,m_k} := 0$ *otherwise. The semiprogram* $P_\phi^{m_1,...,m_k} := \emptyset$.

- *The case for* $\phi \equiv t \leq s$ *is similar.*

- *If* ϕ *is the atomic formula* $t \in_2 A_i$ *then we can as above evaluate* t *and then* $||\phi||^{m_i} := p_{i,\underline{t}}$ *if* $\underline{t} \leq m_i$ *and* $||\phi||^{m_i} := 0$ *otherwise.* $P_\phi^{m_i} := \emptyset$.

- *If* ϕ *is the atomic formula* $A_i \in \mathcal{A}_j$ *then we have two subcases: If* $m_i \leq b_j$ *then* $||\phi||_{b_j}^{m_i} := g_{\mathcal{A}_j}^{b_j}(p_{i,0},...,p_{i,m_i},0,...,0)$. $P_{\phi,b_j}^{m_i} := \emptyset$. *The intention is that* $g_{\mathcal{A}_j}^{b_j}$ *be a free function symbol of arity* b_j *and we shall be careful not to add a definition for any function symbol of this form to our Boolean semiprograms. Furthermore, this is the only case in the construction where a free function symbol is produced.*

  If $m_i > b_j$, *then* $||\phi||_{b_j}^{m_i}$ *is undefined.*

- *If* $\phi \equiv \neg\psi$ *then* $||\phi||_{b_1,...,b_j}^{m_1,...,m_k} := \neg||\psi||_{b_1,...,b_j}^{m_1,...,m_k}$ *and* $P_{\phi,b_1,...,b_j}^{m_1,...,m_k} := P_{\psi,b_1,...,b_j}^{m_1,...,m_k}$.

- *If* $\phi \equiv \psi \circ \xi \quad (\circ \in \{\wedge,\vee\})$, *then* $||\phi||_{b_1,...,b_j}^{m_1,...,m_k} := ||\psi||_{b'_1,...,b'_{j'}}^{m'_1,...,m'_{k'}} \circ ||\xi||_{b''_1,...,b''_{j''}}^{m''_1,...,m''_{k''}}$ *and* $P_{\phi,b_1,...,b_j}^{m_1,...,m_k} := P_{\psi,b'_1,...,b'_{j'}}^{m'_1,...,m'_{k'}} \diamond P_{\xi,b''_1,...,b''_{j''}}^{m''_1,...,m''_{k''}}$. *Here the lists* $\overline{m'}$, $\overline{m''}$, $\overline{b'}$ *and* $\overline{b''}$ *are the sublists of* $\overline{m}$ *and* $\overline{b}$ *corresponding to which of the free variables of* ϕ *occur free in* ψ *and* ξ.

- *If* ϕ *is* $\exists x \leq t\psi(x)$ *then* $||\phi||_{b_1,...,b_j}^{m_1,...,m_k} := \bigvee_{n \leq \underline{t}} ||\psi(n)||_{b_1,...,b_j}^{m_1,...,m_k}$ *(*$\phi(n)$ *is* $\phi(x)[s/x]$ *where* s *is a constant term of value* n, *say* $\overbrace{1+...+1}^{n}$*).* $P_{\phi,b_1,...,b_j}^{m_1,...,m_k} := P_{\psi,b_1,...,b_j}^{m_1,...,m_k}$.

- *If* ϕ *is* $\forall x \leq t\psi(x)$ *then* $||\phi||_{b_1,...,b_j}^{m_1,...,m_k} := \bigwedge_{n \leq \underline{t}} ||\psi(n)||_{b_1,...,b_j}^{m_1,...,m_k}$. $P_{\phi,b_1,...,b_j}^{m_1,...,m_k} := P_{\psi,b_1,...,b_j}^{m_1,...,m_k}$.

- *If* ϕ *is* $\exists X \leq t\psi(X)$ *then* $||\phi||_{b_1,...,b_j}^{m_1,...,m_k} := f_{\phi,b_1,...,b_j}^{m_1,...,m_k}(\overline{\overline{p}})$ *and* $P_{\phi,b_1,...,b_j}^{m_1,...,m_k}$ *is as follows:*

$$f_{\phi,b_1,...,b_j,0}^{m_1,...,m_k,l}(\overline{\overline{p}},q_0,...,q_l) := ||\psi||_{b_1,...,b_j}^{m_1,...,m_k,l} \qquad \textit{for each } l \leq \underline{t}$$

$$f_{\phi,b_1,...,b_j,i}^{m_1,...,m_k,l}(\overline{\overline{p}},q_i,...,q_l) := f_{\phi,b_1,...,b_j,i-1}^{m_1,...,m_k,l}(\overline{\overline{p}},0,q_i,...,q_l) \vee f_{\phi,b_1,...,b_j,i-1}^{m_1,...,m_k,l}(\overline{\overline{p}},1,q_i.,,,.q_l)$$

$$\textit{for } l \leq \underline{t}\ ,\ i \leq l+1$$

$$f^{m_1,\ldots,m_k}_{\phi,b_1,\ldots,b_j}(\overline{\overline{p}}) := \bigvee_{l \le \underline{t}} f^{m_1,\ldots,m_k,l}_{\phi,b_1,\ldots,b_j,l+1}(\overline{\overline{p}})$$

- *If ϕ is $\forall X \le t\psi(X)$ then $||\phi||^{m_1,\ldots,m_k}_{b_1,\ldots,b_j} := f^{m_1,\ldots,m_k}_{\phi,b_1,\ldots,b_j}(\overline{\overline{p}})$ and $P^{m_1,\ldots,m_k}_{\phi,b_1,\ldots,b_j}$ is as follows:*

$$f^{m_1,\ldots,m_k,l}_{\phi,b_1,\ldots,b_j,0}(\overline{\overline{p}}, q_0, \ldots, q_l) := ||\psi||^{m_1,\ldots,m_k,l}_{b_1,\ldots,b_j} \qquad \textit{for each } l \le \underline{t}$$

$$f^{m_1,\ldots,m_k,l}_{\phi,b_1,\ldots,b_j,i}(\overline{\overline{p}}, q_i, \ldots, q_l) := f^{m_1,\ldots,m_k,l}_{\phi,b_1,\ldots,b_j,i-1}(\overline{\overline{p}}, 0, q_i, \ldots, q_l) \wedge f^{m_1,\ldots,m_k,l}_{\phi,b_1,\ldots,b_j,i-1}(\overline{\overline{p}}, 1, q_i., , , .q_l)$$

$$\textit{for } l \le \underline{t}\ ,\ \ i \le l+1$$

$$f^{m_1,\ldots,m_k}_{\phi,b_1,\ldots,b_j}(\overline{\overline{p}}) := \bigwedge_{l \le \underline{t}} f^{m_1,\ldots,m_k,l}_{\phi,b_1,\ldots,b_j,l+1}(\overline{\overline{p}})$$

It is clear that for fixed ϕ, the size of $||\phi||^{m_1,\ldots,m_k}_{b_1,\ldots,b_j}$ is polynomial in $m_1., , , .m_k, b_1, \ldots, b_j$. Whenever we talk of BPLK proofs or BPLK-sequences involving translations of this form, we shall insist that the associated Boolean (semi-)program extend the (semi-)program resulting from the translation.

The following lemma is the main lemma of the proof of Theorem 8.1.1. As discussed earlier, we shall translate sequents with third-order quantifiers as if those third-order variables were free, and then show that BPLK can prove the existence of a function symbol witnessing the sequent in much the same way as in the witnessing theorems from Chapter 7. For this to work it would ordinarily be necessary for the formulas all to be **strict** $\Sigma^{\mathcal{B}}_1$. Unfortunately that cannot be guaranteed since the induction scheme in W^1_1 is for slightly more general formulas. We shall address this problem by first rewriting sequents into the equivalent form given by the replacement theorem and then translating them into the language of Boolean programs.

Lemma 8.1.5. *Let $LK^3 - W^1_1 \vdash \Gamma \longrightarrow \Delta$ where $\Gamma \bigcup \Delta \subset \forall^2\Sigma^{\mathcal{B}}_1$, i.e.*

$$\Gamma = \{\forall A_i \le s_i \exists \mathcal{A}_i \gamma_i(A_i, \mathcal{A}_i, \overline{\mathcal{B}}, \overline{B}, \overline{b})\}$$

and

$$\Delta = \{\forall C_i \le t_i \exists \mathcal{C}_i \delta_i(C_i, \mathcal{C}_i, \overline{\mathcal{B}}, \overline{B}, \overline{b})\},$$

with $\{\gamma_i\} \bigcup \{\delta_i\} \subset \Sigma_0^{\mathcal{B}}$*, and although we write for simplicity the initial string and third-order quantifiers for each formula, in fact for some of the formulas either the initial string quantifier or both initial quantifiers may be absent. Let* $m_1, ..., m_k, n_1, ..., n_l \in \mathbb{N}$ *and let* $b_1, ..., b_j$ *be bounds on the* $\mathcal{A}_i$ *and the* $\overline{\mathcal{B}}$ *such that the translation below is defined. (Sufficient bounds are easily computed by structural induction).*

Then there are function symbols $h_{i,\overline{b}}^{\overline{m},\overline{n}}$ *and BPLK-sequences with endsequents*

$$..., ||\forall A_i \gamma_i(A_i, \mathcal{A}_i^{[A_i]}, \overline{\mathcal{B}}, \overline{B}, \underline{\overline{n}})||_{b_1,...,b_j}^{m_1,...,m_k}, ...$$
$$\longrightarrow ..., ||\forall C_i \delta_i(C_i, \mathcal{C}_i^{[C_i]}, \overline{\mathcal{B}}, \overline{B}, \underline{\overline{n}})||_{b_1,...,b_j}^{m_1,...,m_k}[h_{i,\overline{b}}^{\overline{m},\overline{n}}/g_{\mathcal{C}_i}^{b_i}], ...$$

where $h_{i,\overline{b}}^{\overline{m},\overline{n}}$ *are called witnessing function symbols and are not free, but may be defined in terms of free function symbols (in particular,* $g_{\mathcal{A}_i}^{b_i}$ *). No other free function symbols occur in the BPLK-sequences. These sequences have size polynomial in* $m_1, ..., m_k$*,* $n_1, ..., n_l$ *and* $b_1, ..., b_j$*.*

The notation $...[h_{i,\overline{b}}^{\overline{m},\overline{n}}/g_{\mathcal{C}_i}^{b_i}]$ *in the succedent means that one should perform the translation, substituting function symbol* $h_{i,\overline{b}}$ *for the free symbol* $g_{\mathcal{C}_i}^{b_i}$ *wherever it occurs, including in the definitions of other function symbols* f_ϕ*, which must then be renamed in some consistent way.*

Proof. We begin with an anchored proof in $\mathrm{LK}^3 - W_1^1$ of the sequent in question. We show the existence of the desired BPLK-sequence by induction on the number of sequents in the W_1^1 proof.

Base Case: This is trivial for initial sequents and the witnessing function symbol, if required, is defined to be the constant false predicate (not a "free" function symbol). For translations of axioms B1-B14, L1, L2 and instances of $\Sigma_0^{\mathcal{B}}$-2COMP, it follows from the analogous result for V_1^1 and Extended Frege. For translations of instances of $\Sigma_0^{\mathcal{B}}$-3COMP, the witnessing function symbol has defining formula identical to the (translation of the) comprehension formula, and then the translation of the instance is proved using the introduction rule for this symbol followed by repeated substitutions and $\wedge$: **right**

inferences (to add the universal string quantifier). This symbol will be fully defined (i.e., not "free").

Induction Step: There are cases depending on the final inference of the W_1^1 proof:

1. Weakening, Exchange, introduction of $\neg$, $\vee$ on the right and $\wedge$ on the left:

   These cases are all either structural rules or not applicable to formulas with third-order quantifiers and thus the same rule is applied in the BPLK proof. (A proof of the hypothesis exists by the induction hypothesis). In the case of weakening, the conclusion may have more free variables than the hypothesis. In that case new witnessing function symbols are defined to ignore the new arguments and compute the same values as the old ones, and these must be substituted for the old ones (by induction on the structure of the formula it can easily be seen that a BPLK-sequence can prove each formula equivalent to one with the new function symbols instead).

2. Contraction:, introduction of $\vee$ on the left and $\wedge$ on the right:

   The only obstacle to using the identical propositional rule is that the principal formula of a contraction inference and the side formulas of the two-hypothesis inferences have two ancestors which will in general be witnessed by different witnessing function symbols (if they occur in the succedent). The solution is to define new witnessing function symbols by cases and then for each affected formula prove that the translation witnessed by the new function symbol implies the disjunction of the translations witnessed by the two old symbols.

   For example, a side formula $\forall C_i \leq t_i \exists \mathcal{C}_i \delta_i(C_i, \mathcal{C}_i)$ with witnessing function symbols h_i' and h_i'' would have new witnessing function symbol

$$h_i := (||\delta_i(C_i, \mathcal{C}_i^{[C_i]})||[h_i'/g_{\mathcal{C}_i}] \wedge h_i') \vee (||\delta_i(C_i, \mathcal{C}_i^{[C_i]})||[h_i''/g_{\mathcal{C}_i}] \wedge h_i'')$$

in the conclusion.

3. Introduction of a first-order quantifier:

   These cases are handled by the introduction of the appropriate propositional connective (disjunction or conjunction). In the case of a universal quantifier on the right or of an existential one on the left, proofs for each value of the free variable are concatenated together. In the other cases the proof for the hypothesis is first extended by weakening to add the other disjuncts (conjuncts on the left).

4. Introduction of a second-order quantifier:

   These cases are handled the same way as in the simulation of G by BPLK, in that essentially a big disjunction or conjunction is constructed over all values of a set of propositional variables.

   Additionally, if the principal formula is not $\Sigma_0^{\mathcal{B}}$, the more work is needed. If the principal formula is $\forall C_i \leq t_i \exists \mathcal{C}_i \delta_i(C_i, \mathcal{C}_i)$ in the succedent, then a new witnessing function symbol is defined as follows, to reflect the increased length of $\mathcal{C}_i$, now that it is an array:

   $$h_i'(\overline{p}, \overline{q}) := (\overline{p} = \overline{r} \wedge h_i(\overline{q}))$$

   where $\overline{r}$ are the propositional variables associated with C_i, $\overline{p}$ are precisely as numerous as $\overline{r}$ and $\overline{q}$ are the same variables as the arguments to the original h_i. Then, a derivation is inserted proving

   $$||\delta_i(C_i, \mathcal{C}_i)||[h_i/g_{\mathcal{C}_i}] \longrightarrow ||\delta_i(C_i, \mathcal{C}_i^{[C_i]})||[h_i'/g_{\mathcal{C}_i}].$$

   If the principal formula is $\forall A_i \leq s_i \exists \mathcal{A}_i \gamma_i(A_i, \mathcal{A}_i, \overline{\mathcal{B}}, \overline{B}, \overline{b})$ in the antecedent, then all witnessing function symbols must be modified to supply dummy arguments to $g_{\mathcal{A}_i}$, which now has extra arguments for A_i.

The second-order quantifier introduction is then handled as usual.

5. Introduction of a third-order quantifier:

   These cases are easy: On the left, this amounts to renaming the arguments to the witnessing function symbols (to reflect the possibly new name of the free function symbol) and on the right it means producing a new witnessing function symbol defined equivalent to the existing free function symbol for that variable and substituting it into the sequent.

6. Cut, Induction:

   The cut rule is handled by defining new witnessing function symbols for the conclusion by cases, using the witnessing function symbol for the cut formula (if there is one). Dummy values are also substituted for the free variables present only in the cut formula, and therefore eliminated from the sequent by the cut. Once these symbols are substituted for the original ones, the cut rule may be applied directly.

   For induction this procedure is iterated as many times as the value of the induction bound; a separate proof for each instance of the induction step (as many as the length of the induction) is obtained by the induction hypothesis, and these are concatenated together, following which the procedure for cut is applied repeatedly.

   For example, if the cut formula is $\forall C_i \le t_i \exists \mathcal{C}_i \delta_i(C_i, \mathcal{C}_i)$ and has witnessing function symbol h_i (in the hypothesis where it occurs on the left), then a new witnessing function symbol h_j for $\forall C_j \le t_j \exists \mathcal{C}_j \delta_j(C_j, \mathcal{C}_j)$ would be defined as follows, where h'_j is the witnessing function symbol for the hypothesis with the cut formula on the right, and h''_j that for the hypothesis with the cut formula on the left:

   $$h_j := (||\delta_j(C_j, \mathcal{C}_j^{[C_j]})||[h'_j/g_{\mathcal{C}_j}] \wedge h'_j) \vee h''_j[h_i/g_{\mathcal{C}_j}.$$

   Observe that in the case of cut, $g_{\mathcal{C}_j}$ will not occur in the BPLK-sequence constructed. Therefore in the case of a cut where the conclusion has no third-order variables, the resulting BPLK-sequence will in fact be a BPLK-proof, as the witnessing function symbol h'_j for one hypothesis will be fully defined (not "free"), and therefore so will h_j.

□

8.2 Quantified Boolean Program Proof Systems

In this section we define strong propositional proof systems derived from BPLK by the addition of syntax for quantifying over function symbols, which should be thought of analogously to quantifying over the exponentially large superstrings in theories of third-order bounded arithmetic.

Recall the definition of Boolean programs and the proof system BPLK. In the previous section we extended these definitions to include Boolean semiprograms, allowing "free" (i.e., undefined) function symbols to occur in formulas and function symbol definitions. Now we extend these definitions further. To start, we define a generalization of Boolean programs. The generalized language is defined thus:

Definition 8.2.1. *The language of* **generalized Boolean programs** *(GBPs) consists of the standard propositional language of propositional variables or atoms (p, q, etc.), propositional connectives ($\wedge, \vee, \neg$) and parentheses. Additionally, it includes two sorts of function symbols: First,* **fixed** *function symbols; and second, function symbol* **variables***. Formally there is a distinction between* **free** *and* **bound** *function variables. These will be denoted respectively with f for fixed function symbols, and g and h for free and bound function symbol variables respectively, with various sub- and super-scripts. Finally, it includes quantifiers $\forall$ and $\exists$ (to be applied to bound function symbol variables).*

Now we define a hierarchy of classes of **gbp-formulas** (or simply formulas if clear from the context) over this language. This inductive definition simultaneously defines generalized Boolean programs. Analogously to a standard Boolean program, a generalized Boolean program specifies definitions of fixed function symbols in terms of previously-defined fixed function symbols. Free and bound function variables are intended to be applied to lists of propositional arguments (variables or formulas) and have associated arities. Fixed function symbols, meanwhile, additionally have arguments for function variables, and each therefore has a list of arities: one for each function variable argument, and also a count of propositional arguments.

Definition 8.2.2 (Generalized Boolean Programs and Formula Classes)**.** *A* **Generalized Boolean Program** *P is specified by a finite sequence $\{f_1, ..., f_m\}$ of fixed defined function symbols, where each symbol f_i has an an associated defining equation*

$$f_i(\overline{g_i}; \overline{p_i}) := \phi_i$$

where $\overline{p_i}$ is a list $p_1, ..., p_{k_i}$ of variables, $\overline{g_i}$ is a list of free and bound function symbol variables, and ϕ_i is a semiformula. ϕ_i must be composed entirely of variables among $\overline{p_i}$ and function symbols among $f_1, ..., f_{i-1}$ and $\overline{g_i}$. If ϕ_i is a Σ_j^{bp} (Π_j^{bp})-semiformula, then f_i is a Σ_j^{bp} (Π_j^{bp}) function symbol.

In this context the definition of a **semiformula** *is:*

1. *0,1, and p are $\Sigma_i^{bp} \cap \Pi_i^{bp}$-semiformulas, for any variable p, and all $i \geq 0$.*

2. *Σ_i^{bp} and Π_i^{bp} both contain $\Sigma_{i-1}^{bp} \cup \Pi_{i-1}^{bp}$ for $i > 0$.*

3. *Σ_i^{bp} and Π_i^{bp} are closed under $\wedge$ and $\vee$.*

4. *If ϕ is Σ_i^{bp} (Π_i^{bp}) then $\neg\phi$ is Π_i^{bp} (Σ_i^{bp}).*

5. *If f is a Σ_i^{bp} (Π_i^{bp}) fixed function symbol defined in P, $\psi_1, ..., \psi_k$ are Σ_0^{bp}-***formulas***, and $\overline{g}$ and $\overline{h}$ are lists of free and bound function symbols of the correct arities, then $f(\overline{g}, \overline{h}; \psi_1, ..., \psi_k)$ is a Σ_i^{bp} (Π_i^{bp}) semiformula.*

6. *If g (or h) is a free (respectively, bound) k-ary function symbol variable and $\psi_1,...,\psi_k$ are Σ_0^{bp}-semiformulas, then $g(\psi_1,...,\psi_k)$ (respectively, $h(\psi_1,...,\psi_k)$) is likewise a Σ_0^{bp}-semiformula.*

7. *If ϕ is a Σ_i^{bp}-semiformula and h is a bound function symbol variable, then $\exists h\phi$ is Σ_i^{bp}; if ϕ is Π_i^{bp} then so is $\forall h\phi$. In other words, Σ_i^{bp}-semiformulas are closed under existential function symbol variable quantification, and symmetrically for Π_i^{bp} and universal quantification.*

Finally, if ϕ is a semiformula in which every bound function symbol variable h occurs within the scope of a matching quantifier (i.e. occurs in a sub-semiformula $\exists h\psi$ or $\forall h\psi$ of ϕ), then ϕ is a **formula**.

These classes of gbp-formulas are defined so that the subscript i counts the number of alternations of function symbol quantifiers in a semantic sense (see below for semantics). Consequently, the problem of evaluating a Σ_i^{bp}- or Π_i^{bp}-formula with supplied values for propositional and function symbol variables is in $(\Sigma_i^{exp})^\diamond$ or $(\Pi_i^{exp})^\diamond$, respectively. Conversely, although the definition of the classes excludes several potential ways of forming formulas syntactically (for example, allowing function symbols to be applied only to Σ_0^{bp}-formulas), the translation results below imply that the formula classes are general enough to represent all predicates from the corresponding complexity classes.

We now describe semantics for such formulas. A **generalized truth assignment** σ is a map from propositional variables to $\{T, F\}$ and from free function symbol variables of arity k to k-ary Boolean functions (for each k). A formula ϕ obtains a truth value from an assignment σ inductively as follows:

1. Propositional connectives are as usual.

2. An occurrence of a fixed function symbol is evaluated by first evaluating the Boolean arguments, which must be formulas, and then substituting these truth values and

the supplied free function symbols for the appropriate variables in the defining formula of the function symbol.

3. An occurrence of a free function symbol variable is evaluated by first evaluating the arguments and then applying the Boolean function specified by σ.

4. Finally, a formula $\exists h\phi$ (respectively, $\forall h\phi$) evaluates to true if and only if for some (every) Boolean function of the correct arity, σ, extended to assign this function to the new free function symbol variable g, satisfies the formula $\phi[g/h]$, which is ϕ with every unbound occurrence of h replaced by g.

Sequents constructed from such formulas obtain truth values as usual. A formula or sequent is **valid** if it is satisfied by every truth assignment.

We may now define a PK-like proof system based on gbp-formulas:

Definition 8.2.3. *The system QBP is like the propositional system LK (and therefore includes all rules and initial sequents pf LK), but with the following changes:*

- *Sequents are constructed from gbp-formulas and a proof includes a generalized Boolean program defining all fixed function symbols occurring in the proof.*

- *If the generalized Boolean program contains a definition of the form*

$$f(\overline{g},\overline{p}) := \phi(\overline{g},\overline{p}),$$

the new LK rules f : **left**

$$\frac{\phi(\overline{g'},\overline{\psi}),\Gamma \longrightarrow \Delta}{f(\overline{g'},\overline{\psi}),\Gamma \longrightarrow \Delta}$$

and f : **right**

$$\frac{\Gamma \longrightarrow \Delta,\phi(\overline{g'},\overline{\psi})}{\Gamma \longrightarrow \Delta, f(\overline{g'},\overline{\psi})}$$

*may be used, where $\overline{\psi}$ are precisely as many Σ_0^{bp}-***formulas** *as $\overline{p}$ are variables, and the lists $\overline{g}$ and $\overline{g'}$ of free function symbol variables are the same length and arities.*

- *(***Substitution Rule***) The new inference rule* Σ_0^{bp}**-subst**

$$\frac{\Delta(q,\overline{p}) \longrightarrow \Gamma(q,\overline{p})}{\Delta(\phi,\overline{p}) \longrightarrow \Gamma(\phi,\overline{p})}$$

may be used, where all occurrences of q have been substituted for and ϕ is a Σ_0^{bp}-**formula**.

- *The new rules*

$$\exists : \textbf{left} \quad \frac{\phi(g), \Gamma \longrightarrow \Delta}{\exists h\phi(h), \Gamma \longrightarrow \Delta} \qquad \textit{and} \qquad \forall : \textbf{right} \quad \frac{\Gamma \longrightarrow \Delta, \phi(g)}{\Gamma \longrightarrow \Delta, \forall h\phi(h)}$$

may be used, where g is a free and h a bound function symbol variable of the same arity, and g does not occur in the conclusion of the inference. Furthermore, it must be the case that the $\phi(g)$ is $\phi(h)[g/h]$.

- *The new rules*

$$\forall : \textbf{left} \quad \frac{\phi(f), \Gamma \longrightarrow \Delta}{\forall h\phi(h), \Gamma \longrightarrow \Delta} \qquad \textit{and} \qquad \exists : \textbf{right} \quad \frac{\Gamma \longrightarrow \Delta, \phi(f)}{\Gamma \longrightarrow \Delta, \exists h\phi(h)}$$

where f is **either a free or a fixed function symbol***, and h a bound function symbol variable. Now h must have the same propositional arity (in the case of f fixed, number of propositional arguments) as f. There is no restriction about f occurring in the conclusion (or even in $\exists h\phi(h)$). Also, $\phi(f)$ must be $\phi(h)[f/h]$.*

The system QBP_i is QBP restricted to allow only $\Sigma_i^{bp} \cup \Pi_i^{bp}$-formulas in the cut rule.

Some observations: All the rules are sound, in the sense that they preserve validity. This is clear in the case of the propositional and structural rules. The free/bound distinction on function symbol variables ensures that in the f-introduction, quantifier-introduction and substitution rules, the substitutions in formulas such as $\phi(f)$ or $\phi(\psi)$ never result in any free function symbol variable being caught by a quantifier. Further, any free function symbol variable relevant to the truth-value of a formula ϕ always occurs in ϕ and can never be hidden by defining a fixed function symbol, as these definitions include all free and bound function symbol variables as arguments.

8.3 Propositional Translations of W_1^i

We now show how to translate theorems of W_1^i into (polynomial-size) families of proofs in the proof systems of the last section. We begin with the definition of the propositional translation we consider. This definition is a generalization of the translation defined previously for $\Sigma_0^{\mathcal{B}}$-formulas. We therefore use the same notation $||\cdot||$ to denote it. For the sake of clarity, however, we present the definition in full below. The only differences are the extra cases allowing the direct translation of third-order quantifiers, and the fact that fixed function symbols defined using free function symbol variables now have these variables listed explicitly as arguments.

Definition 8.3.1. *Let $\phi(\mathcal{A}_1, ..., \mathcal{A}_j, A_1, ..., A_k)$ be $g\Sigma_i^{\mathcal{B}}$ in the language $\mathcal{L}_A^3$. For parameters $m_1, ..., m_k, b_1, ..., b_j \in \mathbb{N}$ we construct a generalized Boolean program $P_{\phi,b_1,...,b_j}^{m_1,...,m_k}$ and a formula $||\phi||_{b_1,...,b_j}^{m_1,...,m_k}$ in the language of generalized Boolean programs, with the atoms $\overline{\overline{p}} = (\overline{p_i}, i = 1, ..., k)$, where each $\overline{p_i} = (p_{i,0}, ..., p_{i,m_k})$. The m_i are intended to represent the lengths of the strings in the instance to be translated, while the b_i represent bounds on the lengths of strings contained in the relevant superstring. For some values of the m_i and b_i, the translation will be undefined (when the superstring bounds are too small).*

By induction on the structure of ϕ:

- *If ϕ is the atomic formula $s = t$ then s and t are first-order terms with no free first-order variables. Third-order variables do not appear in first-order terms so all variable occurrences in s and t are of the form $|A_i|$ for some second-order variable A_i. Then using the value m_i for this subterm the terms s and t can be evaluated to $\underline{s}$ and $\underline{t}$. We define $||s = t||^{m_1,...,m_k} := 1$ if $\underline{s} = \underline{t}$ and $||s = t||^{m_1,...,m_k} := 0$ otherwise. The semiprogram $P_\phi^{m_1,...,m_k} := \emptyset$.*
- *The case for $\phi \equiv t \leq s$ is similar.*
- *If ϕ is the atomic formula $t \in_2 A_i$ then we can as above evaluate t and then*

$||\phi||^{m_i} := p_{i,\underline{t}}$ *if* $\underline{t} \leq m_i$ *and* $||\phi||^{m_i} := 0$ *otherwise.* $P^{m_i}_{\phi} := \emptyset$.

- *If* ϕ *is the atomic formula* $A_i \in \mathcal{A}_j$ *then we have two subcases: If* $m_i \leq b_j$ *then* $||\phi||^{m_i}_{b_j} := g^{b_j}_{\mathcal{A}_j}(p_{i,0}, ..., p_{i,m_i}, 0, ..., 0)$. $P^{m_i}_{\phi,b_j} := \emptyset$. *The intention is that* $g^{b_j}_{\mathcal{A}_j}$ *be a function symbol variable of arity* b_j*, and of the free or bound sort according to whether* $\mathcal{A}$ *is. This is the only case in the construction where a function symbol variable is produced.*

  If $m_i > b_j$*, then* $||\phi||^{m_i}_{b_j}$ *is undefined.*

- *If* $\phi \equiv \neg\psi$ *then* $||\phi||^{m_1,...,m_k}_{b_1,...,b_j} := \neg||\psi||^{m_1,...,m_k}_{b_1,...,b_j}$ *and* $P^{m_1,...,m_k}_{\phi,b_1,...,b_j} := P^{m_1,...,m_k}_{\psi,b_1,...,b_j}$.

- *If* $\phi \equiv \psi \circ \xi \quad (\circ \in \{\wedge, \vee\})$*, then* $||\phi||^{m_1,...,m_k}_{b_1,...,b_j} := ||\psi||^{m'_1,...,m'_{k'}}_{b'_1,...,b'_{j'}} \circ ||\xi||^{m''_1,...,m''_{k''}}_{b''_1,...,b''_{j''}}$ *and* $P^{m_1,...,m_k}_{\phi,b_1,...,b_j} := P^{m'_1,...,m'_{k'}}_{\psi,b'_1,...,b'_{j'}} \diamond P^{m''_1,...,m''_{k''}}_{\xi,b''_1,...,b''_{j''}}$. *Here the lists* $\overline{m'}$, $\overline{m''}$, $\overline{b'}$ *and* $\overline{b''}$ *are the sublists of* $\overline{m}$ *and* $\overline{b}$ *corresponding to which of the free variables of* ϕ *occur free in* ψ *and* ξ.

- *If* ϕ *is* $\exists x \leq t\psi(x)$ *then* $||\phi||^{m_1,...,m_k}_{b_1,...,b_j} := \bigvee_{n\leq\underline{t}} ||\psi(n)||^{m_1,...,m_k}_{b_1,...,b_j}$ *(*$\phi(n)$ *is* $\phi(x)[s/x]$ *where* s *is a constant term of value* n*, say* $\overbrace{1 + ... + 1}^{n}$*).* $P^{m_1,...,m_k}_{\phi,b_1,...,b_j} := P^{m_1,...,m_k}_{\psi,b_1,...,b_j}$.

- *If* ϕ *is* $\forall x \leq t\psi(x)$ *then* $||\phi||^{m_1,...,m_k}_{b_1,...,b_j} := \bigwedge_{n\leq\underline{t}} ||\psi(n)||^{m_1,...,m_k}_{b_1,...,b_j}$. $P^{m_1,...,m_k}_{\phi,b_1,...,b_j} := P^{m_1,...,m_k}_{\psi,b_1,...,b_j}$.

- *If* ϕ *is* $\exists X \leq t\psi(X)$ *then* $||\phi||^{m_1,...,m_k}_{b_1,...,b_j} := f^{m_1,...,m_k}_{\phi,b_1,...,b_j}(\overline{g}, \overline{h}; \overline{\overline{p}})$ *and* $P^{m_1,...,m_k}_{\phi,b_1,...,b_j}$ *is as follows:*

$$f^{m_1,...,m_k,l}_{\phi,b_1,...,b_j,0}(\overline{g}, \overline{h}; \overline{\overline{p}}, q_0, ..., q_l) := ||\psi||^{m_1,...,m_k,l}_{b_1,...,b_j} \qquad \text{for each } l \leq \underline{t}$$

$$\begin{aligned} f^{m_1,...,m_k,l}_{\phi,b_1,...,b_j,i}(\overline{g}, \overline{h}; \overline{\overline{p}}, q_i, ..., q_l) &:= f^{m_1,...,m_k,l}_{\phi,b_1,...,b_j,i-1}(\overline{g}, \overline{h}; \overline{\overline{p}}, 0, q_i, ..., q_l) \\ &\vee f^{m_1,...,m_k,l}_{\phi,b_1,...,b_j,i-1}(\overline{g}.\overline{h}; \overline{\overline{p}}, 1, q_i.,,,.q_l) \qquad \text{for } l \leq \underline{t}\ ,\ i \leq l+1 \end{aligned}$$

$$f^{m_1,...,m_k}_{\phi,b_1,...,b_j}(\overline{g}, \overline{h}; \overline{\overline{p}}) := \bigvee_{l\leq\underline{t}} f^{m_1,...,m_k,l}_{\phi,b_1,...,b_j,l+1}(\overline{g}, \overline{h}; \overline{\overline{p}})$$

- *If* ϕ *is* $\forall X \leq t\psi(X)$ *then* $||\phi||^{m_1,...,m_k}_{b_1,...,b_j} := f^{m_1,...,m_k}_{\phi,b_1,...,b_j}(\overline{\overline{p}})$ *and* $P^{m_1,...,m_k}_{\phi,b_1,...,b_j}$ *is as follows:*

$$f^{m_1,...,m_k,l}_{\phi,b_1,...,b_j,0}(\overline{g}, \overline{h}; \overline{\overline{p}}, q_0, ..., q_l) := ||\psi||^{m_1,...,m_k,l}_{b_1,...,b_j} \qquad \text{for each } l \leq \underline{t}$$

$$f^{m_1,\ldots,m_k,l}_{\phi,b_1,\ldots,b_j,i}(\overline{g},\overline{h};\overline{\overline{p}},q_i,\ldots,q_l) := f^{m_1,\ldots,m_k,l}_{\phi,b_1,\ldots,b_j,i-1}(\overline{g},\overline{h};\overline{\overline{p}},0,q_i,\ldots,q_l)$$
$$\wedge\, f^{m_1,\ldots,m_k,l}_{\phi,b_1,\ldots,b_j,i-1}(\overline{g},\overline{h};\overline{\overline{p}},1,q_i.,,,.q_l) \qquad \textit{for } l \le \underline{t}\ ,\ i \le l+1$$

$$f^{m_1,\ldots,m_k}_{\phi,b_1,\ldots,b_j}(\overline{g},\overline{h};\overline{\overline{p}}) := \bigwedge_{l\le\underline{t}} f^{m_1,\ldots,m_k,l}_{\phi,b_1,\ldots,b_j,l+1}(\overline{g},\overline{h};\overline{\overline{p}})$$

- *If* ϕ *is* $\exists\mathcal{X}\psi(\mathcal{X})$ *then* $||\phi||^{m_1,\ldots,m_k}_{b_1,\ldots,b_j} := \exists g^{b_{j+1}}_{\mathcal{X}}||\psi(\mathcal{X})||^{m_1,\ldots,m_k}_{b_1,\ldots,b_{j+1}}$.
- *If* ϕ *is* $\forall\mathcal{X}\psi(\mathcal{X})$ *then* $||\phi||^{m_1,\ldots,m_k}_{b_1,\ldots,b_j} := \forall g^{b_{j+1}}_{\mathcal{X}}||\psi(\mathcal{X})||^{m_1,\ldots,m_k}_{b_1,\ldots,b_{j+1}}$.

Now the translation theorem is:

Theorem 8.3.2. *If* $\phi \in \Sigma^{\mathcal{B}}_i$ *and* $W^i_1 \vdash \phi$ *then* QBP_i *has polynomial-size proofs of the translations* $||\phi||$*; furthermore, these proofs are definable in* S^1_2 *and* V^1 *(or any theory defining polytime functions).*

The proof of this theorem is in fact conceptually simpler than that for the previous translation theorem. This is because the language of the translations is expressive enough to translate the formulas in the proof directly, and so it is not necessary to construct witnessing functions along the way.

Proof. We begin with an anchored proof in $\mathrm{LK}^3 - W^i_1$ of the formula in question. By induction on the number of sequents in the W^i_1 proof, we show that the desired QBP_i proof exists. All formulas in the the W^i_1 proof are $\forall^2\Sigma^{\mathcal{B}}_1$.

Base Case: This is trivial for initial sequents. For translations of axioms B1-B14, L1, L2 and instances of $\Sigma^{\mathcal{B}}_0$-2COMP, it follows from the analogous result for V^1_1 and Extended Frege. Translations of instances of $\Sigma^{\mathcal{B}}_0$-3COMP are proved as follows: A fixed function symbol is defined with defining formula identical to the (translation of the) comprehension formula. The translation of the instance is proved using the introduction rule for this symbol followed by $\exists$: **right** for the function symbol quantifier, then repeated substitutions and $\wedge$: **right** inferences (to add the universal string quantifier).

Induction Step: There are cases depending on the final inference of the W^1_1 proof:

1. Weakening, Exchange, Contraction, introduction of $\neg$, $\vee$, and $\wedge$:

   The same propositional rule is applied.

2. Introduction of a first-order quantifier:

   These cases are handled by the introduction of the appropriate propositional connective (disjunction or conjunction). In the case of a universal quantifier on the right or of an existential one on the left, proofs for each value of the free variable are concatenated together. In the other cases the proof for the hypothesis is first extended by weakening to add the other disjuncts (conjuncts on the left).

3. Introduction of a second-order quantifier:

   These cases are handled the same way as in the simulation of G by BPLK, in that essentially a big disjunction or conjunction is constructed over all values of a set of propositional variables.

4. Introduction of a third-order quantifier:

   The corresponding introduction rule for function symbol variable quantifiers is applied.

5. Cut, Induction:

   The propositional cut rule can be applied directly.

   For induction this is iterated as many times as the value of the induction bound; a separate proof for each instance of the induction step (as many as the length of the induction) is obtained by the induction hypothesis, and these are concatenated together, following which cut is applied repeatedly.

$\square$

Chapter 9

Future Work

In this section we identify some open problems related to the work in this dissertation and its environs. These range from general to specific and are listed below in several categories.

9.1 Specific Questions from this Dissertation

We begin with problems directly arising from the work in this dissertation.

- The slightly unpleasant induction scheme of W_1^1 begs the question: Either prove the replacement theorem of W_1^1 with **strict** $\Sigma_1^{\mathcal{B}}$ induction (i.e., in $\widehat{W_1^1}$), or show this is impossible subject to a complexity assumption as in [22]. The theory HW_1^0, with its universal conservative extension, is a good place to start, as it is possibly amenable to proving a KPT-style witnessing theorem.

- Σ_i^b theorems of T_2^i translate into polynomial size G_i proofs, while such theorems of S_2^i translate into polynomial size G_i^* proofs. By analogy, then, it might be reasonable to expect theorems of TW_1^i and W_1^i to translate into daglike and treelike propositional proofs, respectively. However, the substitution rule seems, on the surface, to be essential in BPLK and QBP$_i$, as it is used in our constructions to form big conjunctions or disjunctions to represent second-order quantifiers. As a result, the treelike versions of these systems might be very weak. Is it is possible that one of the following techniques might allow treelike translations of W_1^i?: First, the use of function-symbol quantifiers to replace some of the string quantifiers; or second, alternative translations in which the universal closure is used instead of free propositional variables, and thus there might be no need of using substitution in this way.

- Related to the above problem is to categorize more precisely the power of the propositional systems: prove their consistency in corresponding theories, and investigate their witnessing problems. There are several degrees of freedom in which to modify the proof systems, such as alternative kinds of substitution, different versions of the rules for introducing function symbols, and so on, and it remains to be seen how these will affect the systems' strength.

- Give a universal theory for exponential time and one for (third-order) polynomial time using the recursion-theoretic characterizations of these classes, in the same style as HW_1^0.

- Are any of the base theories HW_1^0, W_1^1 or TW_1^1 for these classes finitely axiomatizable?

- If W_1^1 is conservative over a minimal theory for PSPACE such as HW_1^0, does this imply any complexity collapse such as is the case for PV and S_2^1? This question applies equally to TW_1^1 and a future minimal theory for exponential time.

9.2 Canonical Proof System For a Complexity Class

It is interesting to note that in some cases, the propositional proof system corresponding to a complexity class has as lines in its proofs objects which are of exactly that complexity class (for example, G, EF, BPLK) yet in other cases, the objects are of seemingly greater computational power (G_1, G_1^*, QBP_i). An interesting open problem is to find, for some of the latter type of examples, a canonical propositional proof system whose lines are exactly the appropriate complexity class. Perhaps a general technique could be devised to deal with many such classes at once.

9.3 Questions About the "Weak Fragments" of Theories and Proof Systems

9.3.1 Relating the Collapse of Theories with the Collapse of Complexity Classes

As discussed in Section 2.1.3, it is plausible that the universal fragments of, for example, W_1^1 and V might be the same without causing any complexity collapse. It would be instructive either to collapse these fragments or to find convincing reasons why it might be impossible.

9.3.2 Collapsing weak fragments of G or QBP

A related issue is that of provability of quantifier-free tautologies in the various subsystems of G and QBP. There does not seem to be any drastic consequence to complexity theory of showing, for example, that G_1 p-simulates G (or even QBP) for such proofs.

9.4 Theories and Proof Systems for Other Complexity Classes

There are many complexity classes for which no corresponding theory or proof system is known. Examples include some NP search classes, but are by no means limited to these. Finding a corresponding theory and proof system and positioning them correctly with respect to already known examples could potentially prove very instructive. For some classes such as the NP search problems already mentioned, there is now a standard technique for producing theories by adding a related axiom to a base theory. In other cases, however, the situation is much more difficult.

Bibliography

[1] *19th IEEE Symposium on Logic in Computer Science (LICS 2004), 14-17 July 2004, Turku, Finland, Proceedings.* IEEE Computer Society, 2004.

[2] M. Ajtai. The complexity of the pigeonhole principle. In *29th Annual Symposium on Foundations of Computer Science*, pages 346–355, White Plains, New York, 24–26 October 1988. IEEE.

[3] S. Buss. *Bounded Arithmetic.* Bibliopolis, Naples, 1986.

[4] Samuel Buss and Jan Krajíček. An application of Boolean complexity to separation problems in bounded arithmetic. *Proceedings of the London Mathematical Society*, 69:1–21, 1994.

[5] Samuel Buss, Jan Krajíček, and Gaisi Takeuti. On provably total functions in bounded arithmetic theories R^i_3, U^i_2 and V^i_2. In Peter Clote and Jan Krajíček, editors, *Arithmetic, proof theory and computational complexity*, pages 116–61. Oxford University Press, Oxford, 1993.

[6] Samuel R. Buss. Axiomatizations and conservation results for fragments of bounded arithmetic. In *CMWLC: Logic and Computation: Proceedings of a Workshop held at Carnegie Mellon University*, pages 57–84. Contemporary Mathematics Volume 106, American Mathematical Society, 1990.

[7] Samuel R. Buss. Relating the bounded arithmetic and polynomial time hierarchies. *Annals of Pure and Applied Logic*, 75(1–2):67–77, 12 September 1995.

[8] Samuel R. Buss, editor. *Handbook of Proof Theory*. Elsevier Science B. V., Amsterdam, 1998.

[9] Samuel R. Buss and Aleksandar Ignjatović. Unprovability of consistency statements in fragments of bounded arithmetic. *Annals of Pure and Applied Logic*, 74:221–244, 1995.

[10] Mario Chiari and Jan Krajíček. Witnessing functions in bounded arithmetic and search problems. *The Journal of Symbolic Logic*, 63(3):1095–1115, September 1998.

[11] Peter Clote. A safe recursion scheme for exponential time. In Sergei I. Adian and Anil Nerode, editors, *LFCS*, volume 1234 of *Lecture Notes in Computer Science*, pages 44–52. Springer, 1997.

[12] Peter Clote and Gaisi Takeuti. Exponential time and bounded arithmetic. In Alan L. Selman, editor, *Structure in Complexity Theory Conference*, volume 223 of *Lecture Notes in Computer Science*, pages 125–143. Springer, 1986.

[13] Alan Cobham. The intrinsic computational difficulty of functions. In Yehoshua Bar-Hillel, editor, *Proceedings of the International Congress for Logic, Methodology and Philosophy of Science*, pages 24–30. North-Holland, 1964.

[14] S. Cook and A. Kolokolova. A second-order system for polytime reasoning using Grädel's theorem. In *16th Annual IEEE Symposium on Logic in Computer Science (LICS '01)*, pages 177–186, Washington - Brussels - Tokyo, June 2001. IEEE.

[15] S. A. Cook. CSC 2429S: Proof Complexity and Bounded Arithmetic. Course notes, URL: "http://www.cs.toronto.edu/∼sacook/csc2429_98", Spring 1998.

[16] S. A. Cook. CSC 2429S: Proof Complexity and Bounded Arithmetic. Course notes, URL: "http://www.cs.toronto.edu/~sacook/csc2429h", Winter 2002.

[17] Stephen Cook and Antonina Kolokolova. A second-order theory for NL. In *LICS* [1], pages 398–407.

[18] Stephen Cook and Tsuyoshi Morioka. Quantified propositional calculus and a theory for NC^1. *Archive for Mathematical Logic*, 2005. To Appear.

[19] Stephen Cook and Phuong Nguyen. Introduction to proof complexity: Bounded arithmetic and propositional translations. Book in Progress.

[20] Stephen Cook and Robert Reckhow. On the lengths of proofs in the propositional calculus (preliminary version). In *Conference Record of Sixth Annual ACM Symposium on Theory of Computing*, pages 135–148, Seattle, Washington, 30 April–2 May 1974.

[21] Stephen Cook and Michael Soltys. Boolean programs and quantified propositional proof systems. *Bulletin of the Section of Logic*, 28(3):119–129, 1999.

[22] Stephen Cook and Neil Thapen. The strength of replacement in weak arithmetic. In *LICS* [1], pages 256–264.

[23] Stephen Cook and Alasdair Urquhart. Functional interpretations of feasibly constructive arithmetic. *Annals of Pure and Applied Logic*, 63(2):103–200, 10 September 1993.

[24] Stephen A. Cook. The complexity of theorem-proving procedures. In *Conference Record of Third Annual ACM Symposium on Theory of Computing*, pages 151–158, Shaker Heights, Ohio, 3–5 1971 1971.

[25] Stephen A. Cook. Feasibly constructive proofs and the propositional calculus (preliminary version). In *Conference Record of Seventh Annual ACM Symposium on Theory of Computing*, pages 83–97, Albuquerque, New Mexico, 5–7 May 1975.

[26] Stephen A. Cook. Relating the provable collapse of P to NC^1 and the power of logical theories. *DIMACS Series in Discrete Math. and Theoretical Computer Science*, 39, 1998.

[27] Stephen A. Cook. Theories for complexity classes and their propositional translations. In Jan Krajíček, editor, *Complexity of Computations and Proofs*, volume 13 of *Quaderni di Matematica*, pages 175–227. Seconda Università di Napoli, 2004.

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

[29] Martin Dowd. *Propositional Representation of Arithmetic Proofs*. PhD thesis, University of Toronto, 1979.

[30] A. Grzegorczyk. Some classes of recursive functions. *Rozprawy Matematyczne*, 4:1–46, 1953.

[31] Petr Hájek and Pavel Pudlák. *Metamathematics of First-Order Arithmetic*. Perspectives in Mathematical Logic. Springer-Verlag, Berlin, 1993.

[32] Armin Haken. The intractability of resolution. *Theoretical Computer Science*, 39(2–3):297–308, August 1985.

[33] Juris Hartmanis. The collapsing hierarchies. *Bulletin of the EATCS*, 33, September 1987.

[34] Aleksandar Ignjatovic. Delineating classes of computational complexity via second order theories with weak set existence principles. I. *The Journal of Symbolic Logic*, 60(1):103–121, March 1995.

[35] Jan Krajíček. On the number of steps in proofs. *Annals of Pure and Applied Logic*, 41:153–78, 1989.

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

[37] Jan Krajíček. On Frege and Extended Frege proof systems. In *P. Clote, J. Remmel (eds.): Feasible Mathematics II*, pages 284–319. Birkhäuser, Boston, 1995.

[38] Jan Krajíček. Implicit proofs. *JSL: Journal of Symbolic Logic*, 69, 2004.

[39] Jan Krajíček and Pavel Pudlák. Propositional proof systems, the consistency of first order theories and the complexity of computations. *The Journal of Symbolic Logic*, 54(3):1063–1079, 1989.

[40] Jan Krajíček and Pavel Pudlák. Quantified propositional calculi and fragments of bounded arithmetic. *Zeitschr. f. Mathematikal Logik u. Grundlagen d. Mathematik*, 36:29–46, 1990.

[41] Jan Krajíček, Pavel Pudlák, and Gaisi Takeuti. Bounded arithmetic and the polynomial hierarchy. *Annals of Pure and Applied Logic*, 52(1–2):143–153, 1991.

[42] Jan Krajíček and Gaisi Takeuti. On bounded Σ_1^1 polynomial induction. In S. R. Buss and P. J. Scott, editors, *FEASMATH: Feasible Mathematics: A Mathematical Sciences Institute Workshop*, pages 259–80. Birkhauser, 1990.

[43] Stanisław Leśniewski. Grundzüge eines neunen Systems der Grundlagen der Mathematik. *Fundamenta Mathematicae*, 14:1–81, 1929.

[44] Phuong Nguyen and Stephen Cook. VTC^0: A second-order theory for TC^0. In *LICS* [1], pages 378–387.

[45] Phuong Nguyen and Stephen Cook. Theories for TC^0 and other small complexity classes. *Logical Methods in Computer Science*, 2005. To Appear.

[46] J. Paris and A. Wilkie. Counting problems in bounded arithmetic. In *Methods in Mathematical Logic*, volume 1130 of *LNM*, pages 317–40. Springer-Verlag, 1985.

[47] J. Paris and A. Wilkie. On the scheme of induction for bounded arithmetic formulas. *Annals of Pure and Applied Logic*, 35:261–302, 1987.

[48] Pavel Pudlák. A note on bounded arithmetic. *Fundamenta Mathematica*, 136:85–9, 1990.

[49] Alexander A. Razborov. An equivalence between second order bounded domain bounded arithmetic and furst order bounded arithmetic. In Peter Clote and Jan Krajíček, editors, *Arithmetic, proof theory and computational complexity*, pages 247–77. Oxford University Press, Oxford, 1993.

[50] Alexander A. Razborov. Bounded arithmetic and lower bounds in complexity. In P. Clote and J. Remmel, editors, *Feasible Mathematics II*, pages 344–386. Birkhauser, 1995.

[51] Alexander A. Razborov and Steven Rudich. Natural proofs. *Journal of Computer and System Sciences*, 55(1):24–35, August 1997.

[52] R. W. Ritchie. Classes of predictably computable functions. *Transactions of the American Mathematical Society*, 106:139–173, 1963.

[53] Alan Skelley. Relating the PSPACE reasoning power of Boolean programs and quantified Boolean formulas. Master's thesis, University of Toronto, 2000. Available from ECCC in the 'theses' section.

[54] Alan Skelley. Propositional PSPACE reasoning with Boolean programs versus quantified Boolean formulas. In *ICALP*, volume 3142 of *Lecture Notes in Computer Science*, pages 1163–1175. Springer, 2004.

[55] Alan Skelley. A third-order bounded arithmetic theory for PSPACE. In Jerzy Marcinkowski and Andrzej Tarlecki, editors, *CSL*, volume 3210 of *Lecture Notes in Computer Science*, pages 340–354. Springer, 2004.

[56] Michael Soltys. A model-theoretic proof of the completeness of LK proofs. Manuscript, available on author's web page, 1999.

[57] Gaisi Takeuti. RSUV isomorphism. In Peter Clote and Jan Krajíček, editors, *Arithmetic, proof theory and computational complexity*, pages 364–86. Oxford University Press, Oxford, 1993.

[58] D. Zambella. Notes on polynomially bounded arithmetic. *The Journal of Symbolic Logic*, 61(3):942–966, 1996.

[59] D. Zambella. End extensions of linearly bounded arithmetic. *Annals of Pure and Applied Logic*, 88, 1997.