Bounded Arithmetic

Samuel R. Buss
Department of Mathematics
University of California, Berkeley

TABLE OF CONTENTS

ACKNOWLEDGEMENTS

This book is a reprinting of my Ph.D. dissertation submitted to the Department of Mathematics at Princeton University in May 1985; a number of minor corrections have been made.

I benefited greatly from frequent conversations with my advisor, Simon Kochen. He helped me learn mathematical logic by giving me a "guided tour" of the literature and sparked my interest in formal theories of arithmetic.

I wish to thank Ed Nelson for sharing his insights into predicative arithmetic and for giving me access to his unpublished work.

In addition, Richard Lipton, Alex Wilkie, Pavel Pudlak, Steve Cook, Harvey Friedman and Martin Dowd each made suggestions or asked questions which led to additions to this dissertation.

I am grateful to John Lafferty for proofreading a preliminary copy of the dissertation and an earlier extended abstract.

John Doner, Bob Solovay, Gaisi Takeuti and Osamu Watanabe made suggestions for corrections to the original version of the dissertation which have been made in this reprinting.

My greatest debt is to my wife, Teresa, who patiently lived through my dissertation and provided much support, emotional and otherwise. She also helped with proofreading and by writing computer programs to aid in the typesetting. This dissertation is dedicated to her.

The National Science Foundation and the Sloane Foundation partially supported me while this dissertation was being written. Princeton University supplied computer facilities for the original typing and printing of this dissertation and the Mathematical Sciences Research Institute provided facilities to re-typeset my dissertation in book format.

Introduction

The fundamental questions of theoretical computer science ask what are the most efficient methods to compute a given function. A variety of computational models are used including the Chomsky hierarchy, time and/or space bounded Turing machines, alternating Turing machines, array processors and many others. The functions or decision problems considered by computer scientists are almost always combinatorial or numerical in nature.

Mathematical logic has also long studied problems in computability theory. However the aims and scope of mathematical logic and computational complexity have been quite different. Classically, mathematical logic has considered general recursive functions as its principal model for computability, whereas computer science likes to deal with functions which are actually computable in the real world. Mathematical logic has rarely considered classes of functions simpler than the primitive recursive functions, while computer science seldom treats problems which are not elementary recursive in the sense of Kalmar.

However, the problems of theoretical computer science can often be stated in terms familiar to mathematical logic. For concreteness, suppose we are given a function f. Frequently we can, without loss of generality, reduce f to a decision problem. By suitably encoding instances of the decision problem we can reduce the problem of computing f to the problem of recognizing a formal language Λ_f. Now we can show that f is computable (relative to a given model of computation) if and only if the language Λ_f is definable in a certain formal way (which obviously depends on the model of computation). Thus we have restated a question about the computability of f as a question about the definability of Λ_f.

Questions about the most efficient or simplest means of defining an object have long been considered by mathematical logic. For instance, quantifier elimination has been investigated for many formal systems. Thus the problem of how the formal language Λ_f can be defined may legitimately be considered part of mathematical logic.

This dissertation uses methods from mathematical logic to examine issues related to computational complexity. The kind of question dealt with is as follows: Given a formal theory R, what functions can R define? Or, what function symbols may be introduced in R?

We say that R can define a function f when R proves $(\forall x)(\exists! y)A(x,y)$ and f is defined to satisfy $A(x,f(x))$ for all x. In other words, a proof of $(\forall x)(\exists! y)A(x,y)$ provides an implicit definition of the function $y=f(x)$.

A *constructive* proof of $(\forall x)(\exists y)A(x,y)$ by definition contains an algorithm for computing f. Thus a constructive proof gives us an effective way (at least in principle) to compute f; that is to say, a constructive proof specifies a recursive algorithm to compute y from x.

However, a recursive function may be computable only in a theoretical sense: the time required to compute it may be far larger than the lifespan of the universe. We are more interested in *feasibly computable* functions, which can be calculated by today's (or tomorrow's) computers. It is generally accepted that the correct formal definition for a *feasible* function is that the function be computable in polynomial time; i.e., that the runtime of some Turing machine computing the function be bounded by a polynomial in the length of the input.

Accordingly, we are interested in the question of when the existence of a proof of $(\forall x)(\exists y)A(x,y)$ implies the existence of a feasible algorithm which, given x, computes y. A natural condition to put on a proof is that it be a valid proof of a certain formal theory (indeed this is unavoidable). We can also put conditions on the formula A. The main results of this dissertation show that certain restrictions of these types on a proof of $(\forall x)(\exists y)A(x,y)$ imply the existence of a function f such that $(\forall x)A(x,f(x))$ and such that f has a certain computational complexity. In particular, we may be able to deduce that f is polynomial time computable, f is at a certain level of the polynomial hierarchy, f is polynomial space computable, or f is exponential time computable.

We shall discuss exclusively a family of formal theories called *Bounded Arithmetic*, which are weak fragments of Peano arithmetic. The language of Bounded Arithmetic includes the following function and predicate symbols:

0	zero constant symbol
S	successor
$+$	addition
$\cdot$	multiplication
$\lfloor \frac{1}{2}x \rfloor$	"shift right" function, i.e., divide by two and round down
$\lvert x\rvert$	$=\lceil \log_2(x+1)\rceil$, the length of the binary representation of x
$x\#y$	$=2^{\lvert x\rvert\cdot\lvert y\rvert}$, the "smash" function
$\leq$	less than or equal to

(The notations $\lfloor a\rfloor$ and $\lceil a\rceil$ denote the greatest integer $\leq a$ and the least integer $\geq a$.)

In Bounded Arithmetic, quantifiers of the form $(\forall x)$ or $(\exists x)$ are called *unbounded* quantifiers. We also use *bounded* quantifiers which are of the form $(\forall x\leq t)$ or $(\exists x\leq t)$ where t is any term not involving x. The meanings of $(\forall x\leq t)A$ and $(\exists x\leq t)A$ are $(\forall x)(x\leq t\supset A)$ and $(\exists x)(x\leq t\wedge A)$, respectively. A formula is *bounded* if and only if it contains no unbounded quantifiers. The principal difference between Bounded Arithmetic and Peano arithmetic is that in theories of Bounded Arithmetic the induction axioms are restricted to bounded formulae.

A special kind of bounded quantifiers are the *sharply bounded quantifiers*, which are those of the form $(\forall x\leq|t|)$ or $(\exists x\leq|t|)$, where t is a term not involving x. We classify the bounded formulae in a hierarchy Σ_0^b, Σ_1^b, Π_1^b, Σ_2^b, Π_2^b, $\cdots$ by counting alternations of bounded quantifiers, ignoring the sharply bounded quantifiers. This is analogous to the definition of the arithmetic hierarchy since formulae are classified in the arithmetic hierarchy by counting alternations of unbounded quantifiers, ignoring bounded quantifiers. Hence, in Bounded Arithmetic, the roles of bounded and sharply bounded quantifiers are analogous to the roles of unbounded and bounded quantifiers, respectively, in Peano arithmetic.

The most important axioms for Bounded Arithmetic are the induction axioms. The induction axioms are restricted to certain subsets of the bounded formulae. We are most interested in a modified induction axiom called Σ_i^b*–PIND*. The Σ_i^b*–PIND* axioms are the formulae

$$A(0)\wedge(\forall x)(A(\lfloor \tfrac{1}{2}x\rfloor)\supset A(x))\supset(\forall x)A(x)$$

where A is a Σ_i^b–formula. We define in Chapter 2 a hierarchy of theories S_2^0, S_2^1, $S_2^2, \ldots$ so that S_2^i is a theory of Bounded Arithmetic axiomatized by a few simple open axioms and by Σ_i^p*–PIND*.

If R is a theory of Bounded Arithmetic we say that the function f is Σ_i^b–definable in R iff there is a Σ_i^b–formula $A(x,y)$ such that

(a) For all x, $A(x,f(x))$ is true.
(b) $R\vdash(\forall x)(\exists y)A(x,y)$
(c) $R\vdash(\forall x)(\forall y)(\forall z)(A(x,y)\wedge A(x,z)\supset y=z)$

We shall be mostly interested in functions which are Σ_i^b–definable in S_2^i.

The Meyer-Stockmeyer polynomial hierarchy is a hierarchy of predicates on the nonnegative integers which can be computed in polynomial time by a generalized version of a Turing machine. The smallest class of the polynomial hierarchy is P, the set of predicates computable in polynomial time by some Turing machine. One step up is the class Σ_1^p, or NP, the set of predicates computable by a non-deterministic polynomial time Turing machine. It is an important open question whether $P=NP$. The classes in the polynomial hierarchy are P, Σ_1^p, Π_1^p, Σ_2^p, $\Pi_2^p, \ldots$

We can extend the polynomial hierarchy to a hierarchy of functions by defining $\Box_{i+1}^p = PTC(\Sigma_i^p)$, the *Polynomial-Time Closure* of Σ_i^p, to be the set of functions which can be computed by a polynomial time Turing machine (i.e. a transducer) with an oracle for a predicate in Σ_i^p.

It is well known (and we prove it again in Chapter 1) that the predicates in Σ_i^p are precisely the predicates which can be expressed by a Σ_i^b–formula. This fact provides a link between computational complexity and the quantifier structure of formulae.

The principal theorem of this dissertation states that any function which is Σ_i^b–definable in S_2^i is a $\Box_i^p$–function, and conversely that every $\Box_i^p$–function is Σ_i^b–definable in S_2^i. (See Theorem 5.6 for the strongest version of this theorem.) This provides a characterization of the functions which are Σ_i^b–definable in S_2^i in terms of computational complexity.

The hardest part of this theorem is showing that every Σ_i^b–definable function is in $\square_i^p$. An extremely brief outline of the proof is as follows: Let A be a Σ_i^b–formula and suppose S_2^i proves $(\forall x)(\exists y)A(x,y)$. By Gentzen's cut elimination theorem there is a free cut free proof of $(\forall x)(\exists y)A(x,y)$. By examining the allowable inferences of natural deduction we discover that this free cut free proof contains an explicit $\square_i^p$–algorithm for computing y from x. This method of proof is reminiscent of Kreisel [18] and Goad [14], in that one of the important ideas is that a free cut free proof can be "unwound" to yield an algorithm. The proof is carried out in detail in Chapter 5.

A corollary to the main theorem is that S_2^1 can Σ_1^p–define precisely the polynomial time functions and S_2^2 can Σ_2^b–define precisely the functions in $PTC(NP)$.

The import of this theorem is twofold. On one hand, it provides a characterization of the $\square_i^p$–functions in terms of their definability by the formal theory S_2^i of arithmetic. On the other hand, it states that the proof-theoretic strength of the formal theory S_2^i is closely linked to the computational complexity of $\square_i^p$–functions.

Another way to state the main theorem is as follows: if $A \in \Sigma_i^b$ and $B \in \Pi_i^b$ and if $S_2^i \vdash A \leftrightarrow B$, then the predicate defined by A and B is in $PTC(\Sigma_{i-1}^p)$. In particular, any predicate which S_2^1 can prove is equivalent to both a Σ_1^b– and a Π_1^b–formula is in P; in other words, since Σ_1^b– and Π_1^b–formulae represent NP and co–NP predicates, the class of predicates which S_2^1 proves are in $NP \cap$co–NP is the class P of polynomial time predicates. (It is an open question whether $NP \cap$co–NP is equal to P.)

In Chapters 9 and 10 we discuss second-order theories of Bounded Arithmetic. We define two theories U_2^1 and V_2^1 of second-order Bounded Arithmetic which have the property that the functions $\Sigma_1^{1,b}$–definable in U_2^1 (respectively, V_2^1) are precisely the functions which are computable by some polynomial space Turing machine (respectively, by some exponential time Turing machine). This provides a characterization of the PSPACE and EXPTIME functions in terms of definability in second-order Bounded Arithmetic.

Chapter 7 discusses improved versions of Gödel incompleteness theorems for Bounded Arithmetic. It is shown that the theory S_2^1 is strong enough to carry out the arithmetization of metamathematics. Thus there is a formula $FCFCon(S_2^i)$ which asserts that there is no free cut free S_2^i–proof of a contradiction. Also, there is a formula $BDCon(S_2^i)$ which asserts that there is no S_2^i–proof P of a contradiction such that every formula in P is bounded. We show that, for $i \geq 1$, S_2^i can not prove either $FCFCon(S_2^i)$ or $BDCon(S_2^i)$.

One of our most important open questions is whether the hierarchy of theories S_2^1, S_2^2, $S_2^3, \ldots$ is proper. Of course this is analogous to the open problem of whether the polynomial hierarchy is proper. In Chapter 7 we make an unsuccessful attempt to prove that this hierarchy of theories is proper.

Chapter 8 builds upon the work of Chapter 7; the main theorem of Chapter 8 is a restatement of the $NP = ?$ co–NP problem in proof-theoretic terms. It turns out that NP is equal to co–NP iff there is a bounded theory R of arithmetic satisfying a certain "anti-reflection" property. See Theorem 8.6 for the precise statement.

The prerequisites for reading this dissertation are some knowledge of computational complexity and of first order logic. Garey & Johnson [12] is a good introduction to computational complexity; in addition, the polynomial hierarchy is defined in detail in Chapter 1 below. Takeuti [28] is the best source for the proof theory that we use; in particular, our treatment of the cut elimination theorem is taken directly from Takeuti. For the reader who has studied first order logic but not proof theory, Chapter 4 has an introduction to proof theory and the cut elimination theorem.

Chapter 1

The Polynomial Hierarchy

This first chapter defines the polynomial hierarchy and explains the link between the computer science definition and the mathematical logic definition. We begin by defining the polynomial hierarchy by using limited iteration and we prove that this definition is equivalent to the usual definition in terms of Turing machines. We then discuss how the polynomial hierarchy can be defined without using limited iteration. The main result of interest to us is Theorem 8 which states that the polynomial hierarchy corresponds to a hierarchy of bounded formulae of Bounded Arithmetic.

The results of this chapter are equivalent to the original work of Cobham [5], Stockmeyer [26] and Wrathall [33], but they are stated and proved in a different form. Some of the results are due originally to Kent-Hodgson [17].

1.1. Limited Iteration.

An important class of functions is the class of functions which can be computed in polynomial time. By polynomial time, we mean that the number of steps in some program which computes the function is bounded by a polynomial of the *length* of the input. The concept of polynomial time is invariant for Turing machines and modern day sequential programming languages, as well as for other models of computation such as Random Access Machines (RAM's). For example, if a RAM program runs in time $p(n)$ on inputs of length n, a multitape Turing machine can simulate the action of the RAM program in time $O((p(n))^2)$, (see [1]). Hence if $p(n)$ is bounded by a polynomial, so is the running time of the Turing machine.

Instead of defining polynomial time computations directly in terms of Turing machines, we will define an operation called *limited iteration* for obtaining new functions. By starting with a base set of functions and taking its closure under composition and limited iteration, we can construct all polynomial time computable functions.

We adopt the convention that all functions have domain $\mathbf{N}^k$ and codomain $\mathbf{N}$ for the rest of this dissertation where $\mathbf{N}$ denotes the natural numbers. Another approach which is often used is that functions have domain and range the set of strings of symbols from a finite alphabet. These two approaches are essentially equivalent; indeed, an integer can be considered as a string of zeros and ones, namely as its binary representation. However we find it advantageous to use integers since it allows us to relate the polynomial hierarchy to formal theories of arithmetic (in later chapters).

Definition: B is the following set of functions from $\mathbf{N}^k$ to $\mathbf{N}$:

(1) 0, (the constant zero function)

(2) $x \mapsto Sx$, (the successor function)

(3) $x \mapsto \lfloor \frac{1}{2}x \rfloor$, (the shift right function)

(4) $x \mapsto 2 \cdot x$, (the shift left function)

(5) $(x,y) \mapsto x \leq y \;=\; \begin{cases} 1 & \text{if } x \leq y \\ 0 & \text{if } x > y \end{cases}$

(6) $(x,y,z) \mapsto Choice(x,y,z) \;=\; \begin{cases} y & \text{if } x > 0 \\ z & \text{if } x = 0 \end{cases}$

B will be the base set of functions from which we will obtain the polynomial time functions. The first operation we can use to obtain new functions is composition. Composition is best defined by a few examples:

Examples:

(1) *Logical operations.* We will use the conventions that if $x>0$ then x represents *True* and if $x=0$ then x represents *False*.

Negation: $(\neg x) \;=\; x \leq 0 \;=\; Choice(x,0,1)$
And: $(x \wedge y) \;=\; Choice(x,y,0)$
Or: $(x \vee y) \;=\; Choice(x,1,y)$
Xor: $(x \oplus y) \;=\; (\neg x \wedge y) \vee (x \wedge \neg y)$

It is important to note that for the time being $\neg$, $\wedge$, $\vee$ and $\oplus$ are numerical operations. Later we will use $\neg$, $\wedge$ and $\vee$ extensively as logical operators.

(2) *Equality and Inequality:*

$$(x \equiv y) \;=\; (x \leq y) \wedge (y \leq x)$$

$$(x < y) \;=\; (x \leq y) \wedge \neg(x \equiv y)$$

(3) *Arithmetic modulo 2:*

$$(x\%2) \;=\; \neg(x \equiv 2 \cdot \lfloor \tfrac{1}{2}x \rfloor)$$

$(x\%2)$ is equal to zero if x is even and one if x is odd.

We also need to define functions for handling finite sequences of numbers. We will code our sequences by values called Gödel numbers. The Gödel number for the sequence $a_1, a_2, \ldots, a_k$ is constructed as follows. First write the a_i's in binary notation so we have a string of 0's, 1's and commas. Then write the string in reverse order and replace each 0 by "10", each 1 by "11" and each comma by "00". The resulting string of zeros and ones is the binary representation of the Gödel number $<a_1, \ldots, a_k>$. For example the Gödel number of 3,4,5 is $(11101100101011001111)_2$ or 969,423. The empty sequence has Gödel number $<>=0$.

Definition: B^+ is the set of functions which contains all the functions in B plus the following functions:

(1) $\beta(i,<a_1, \ldots, a_n>) = \begin{cases} n & \text{if } i=0 \\ a_i & \text{if } 0<i\leq n \end{cases}$

The value of β may be defined arbitrarily when the second argument is not a valid Gödel number for a sequence or if $i>n$.

(2) $Truncate(<a_1,a_2, \ldots, a_n>) = <a_2, \ldots, a_n>$

(3) $a_0 * <a_1, \ldots, a_n> = <a_0, a_1, \ldots, a_n>$

Again, the values of the functions *Truncate* and $*$ have not been specified for arguments which are not Gödel numbers of sequences; it makes no difference how they are defined for arguments other than those above.

Definition: We define the unary function $|x|$ to be $\lceil \log_2(x+1) \rceil$, or the length of the binary representation of x. Note that $|0|=0$.

If $\vec{x}$ is a vector of numbers $x_1, \ldots, x_n$ then $|\vec{x}|$ denotes the vector $|x_1|, \ldots, |x_n|$.

Definition: p is a *suitable* polynomial iff p has nonnegative integer coefficients.

Definition: Let $k\geq 0$ and let $g:\mathbf{N}^k \longrightarrow \mathbf{N}$ and $h:\mathbf{N}^{k+2} \rightarrow \mathbf{N}$ be arbitrary functions and let p and q be suitable polynomials. We say that $f:\mathbf{N}^k \rightarrow \mathbf{N}$ is *defined by limited iteration from g and h with time bound p and space bound q* iff the following holds:

Let $\tau:\mathbf{N}^{k+1} \rightarrow \mathbf{N}$ be defined as

$$\begin{aligned} \tau(x_1, \ldots, x_k, 0) &= g(x_1, \ldots, x_k) \\ \tau(x_1, \ldots, x_k, n+1) &= h(x_1, \ldots, x_k, n, \tau(x_1, \ldots, x_k, n)). \end{aligned}$$

Then we must have

$$(\forall n \leq p(|\vec{x}|))(|\tau(\vec{x},n)| \leq q(|\vec{x}|))$$

and $f(\vec{x})$ is defined by

$$f(\vec{x}) = \tau(\vec{x}, p(|\vec{x}|)).$$

Our definition for limited iteration is very similar to what Grzegorczyk [15] and Cobham [5] call "limited recursion".

Definition: A function $f{:}\mathbf{N}^k \to \mathbf{N}$ has *polynomial growth rate* iff there is a suitable polynomial p such that for all $\vec{x}$, we have $|f(\vec{x})| \le p(|\vec{x}|)$. Let C be a set of functions of polynomial growth rate. The *Polynomial-time closure* of C, $PTC(C)$, is the smallest class of functions which (1) contains C and B and (2) is closed under composition and definition by limited iteration.

Theorem 1: $PTC(\emptyset) \supseteq B^+$.

Proof: This is a technical result and the proof is in the appendix to this chapter. □

As an illustration of how limited recursion is used, we show that addition is in $PTC(\emptyset)$. We first define $f_1(x,y)$ by limited recursion from g_1 and h_1 with bounds p_1 and q_1, where

$$\begin{aligned}
g_1(x,y) &= 1{*}0{*}x{*}y{*}0 = \langle 1,0,x,y \rangle \\
h_1(x,y,m,w) &= SM(\beta(1,w),\beta(2,w),\beta(3,w)\%2,\beta(4,w)\%2){*} \\
&\qquad {*}CARRY(\beta(2,w),\beta(3,w)\%2,\beta(4,w)\%2){*} \\
&\qquad {*}\lfloor \tfrac{1}{2}\beta(3,w) \rfloor {*} \lfloor \tfrac{1}{2}\beta(4,w) \rfloor {*}0 \\
p_1(n,m) &= n+m \\
q_1(n,m) &= 2n+2m+14
\end{aligned}$$

and where

$$\begin{aligned}
SM(x,a,b,c) &= Choice(a \oplus b \oplus c, S(2\cdot x), 2\cdot x) \\
CARRY(a,b,c) &= (a \wedge b) \vee (a \wedge c) \vee (b \wedge c).
\end{aligned}$$

Note that in the definition of g_1, the formula $1{*}0{*}x{*}y{*}0$ means $1{*}(0{*}(x{*}(y{*}0)))$ which is $\langle 1,0,x,y \rangle$. Similar considerations apply to the definition of h_1 and for the rest of Chapter 1 we follow the convention that $*$ associates from right to left.

Intuitively, $f_1(x,y) = \langle FlippedSum(x,y),0,0,0 \rangle$, where $FlippedSum(x,y)$ is a number whose binary expansion contains the binary expansion of $x+y$ in *reverse* order immediately following the high order bit. For example, $f_1(4,8) = \langle (10011000)_2,0,0,0 \rangle$. Since g_1 and h_1 are defined by composition from functions in B^+, Theorem 1 says that g_1, $h_1 \in PTC(\emptyset)$. Hence $f_1 \in PTC(\emptyset)$.

Secondly, we define $f_2(x)$ by limited iteration from g_2 and h_2 with bounds p_2 and q_2, where

$$
\begin{aligned}
g_2(x) &= 0{*}x{*}0 = \langle 0,x \rangle \\
h_2(x,m,w) &= \mathit{Choice}(\beta(2,w)\%2, S(2\cdot\beta(1,w)), 2\cdot\beta(1,w)){*}\lfloor \tfrac{1}{2}\beta(2,w)\rfloor{*}0 \\
p_2(n) &= n \\
q_2(n) &= 2n+6.
\end{aligned}
$$

We now define $\mathit{Flip}(x)$ using composition by

$$\mathit{Flip}(x) = \beta(1, f_2(x))$$

and finally we can define addition as

$$x+y = \lfloor \tfrac{1}{2}\mathit{Flip}(\beta(1, f_1(x,y)))\rfloor.$$

1.2. Polynomial-time Computations.

In this section, we show that the operation of limited iteration can be used to define the concept of polynomial time computation.

Theorem 2: Let C be a set of functions with polynomial growth rate. Then $f \in PTC(C)$ iff there is a finite set $\{h_1, \ldots, h_k\} \subseteq C$ and a Turing machine M_f with oracles for $h_1, \ldots, h_k$ so that M_f computes f in polynomial time.

Note that we are allowing M_f to use oracles for *functions* h_i. In order to be defined properly it is required that when the oracle is consulted, the elapsed time reflect the length of the input to and/or the output from the oracle. Garey and Johnson [12] define this concept as Oracle Turing machines with a correction to the definition at the end of their book (the first edition). Another way to define function oracles is to count an oracle invocation as a simple time unit and to put an a priori restriction on the amount of space used by the Turing machine. Thus if we limit both the time and the space we get a correct definition of a Turing machine which uses function oracles.

Definition: P is the set of functions computable by polynomial time Turing machines.

Corollary 3: $PTC(\emptyset) = P$.

Proof: of Theorem 2.

$\Longrightarrow$ First we show that $f \in PTC(C)$ implies that the desired M_f exists. The proof is by induction on the complexity of the definition of f. To start the proof by induction we note that if f is in $B \cup C$ the result is obvious. If f is defined by composition from functions in $PTC(C)$ the induction step is easy.

Suppose f is defined by limited iteration from g and h with time bound p and space bound q. The induction hypothesis is that there are Turing machines M_g and M_h which compute g and h and have runtimes bounded by suitable polynomials p_g and p_h respectively. Let M_f be the Turing machine which uses M_g and M_h as "subprograms" to compute f in a straightforward manner. Then the runtime of M_f is approximately bounded by

$$p_g(|\vec{x}|)+p(|\vec{x}|)\cdot p_h(|\vec{x}|,p(|\vec{x}|),q(|\vec{x}|)).$$

This bound is approximate since it does not provide for the overhead of M_f invoking M_g and M_h; however, clearly M_f is polynomial time.

$\Longleftarrow$ Let M be a polynomial-time Turing machine with oracles $h_1,\ldots,h_k \in C$ and runtime bounded by the polynomial p. Let $q(n)$ be a polynomial bounding the total amount of tape space used by M on inputs of length n. We want to show that the function M computes is in $PTC(C)$. Let the states of M be $q_0,\ldots,q_{N+k}$ where q_0 is the initial state and q_{N+i} is the oracle state for h_i. We assume without loss of generality that M has two tapes with alphabet $b_0,\ldots,b_J$ where $J \geq 2$ and b_0 is the blank symbol. An ID (instantaneous description) of M is given by the following items:

(1) The contents of the work tape (current head position is at b_{t_0}):

$$b_{t_{-l}} \cdots b_{t_{-1}}\, b_{t_0}\, b_{t_1} \cdots b_{t_r}.$$

(2) The contents of the oracle tape (current head position is at b_{s_0}):

$$b_{s_{-m}} \cdots b_{s_{-1}}\, b_{s_0}\, b_{s_1} \cdots b_{s_n}.$$

(3) The current state q_u.

We assume that the input and output of M are coded as a binary string with b_1 coding 0 and b_2 coding 1. M is presumed to start with the worktape positioned on the leftmost bit of the input and to halt on the leftmost bit of the output. The inputs and outputs for the oracles are coded similarly. The convention for invoking an oracle is that upon entering state q_{N+i}, the oracle for h_i is invoked with input value coded by the string $b_{s_0} \cdots b_{s_n}$; the value output by the oracle is coded as a binary string and written on the oracle tape as the string $b_{s_0} \cdots b_{s_{\bar{n}}}$. After invoking an oracle the next state M enters is q_N.

We will code an ID of M by (the Gödel number of) the sequence

$$<u,<t_{-1},\ldots,t_{-l}>,<t_0,\ldots,t_r>,<s_{-1},\ldots,s_{-m}>,<s_0,\ldots,s_n>>.$$

We define f by the following procedure: we first define functions *Init*, *Next*, and *Decode*, then define f_3 by limited iteration from *Init* and *Next*, and finally define $f(x) = Decode(\beta(3,f_3(x)))$.

Init is the function computing the initial state of M with input x. We first define $f_1(x)$ by limited iteration from g_1 and h_1 with bound p_1 and q_1, where

$$\begin{aligned} g_1(x) &= <0,x> \\ h_1(x,m,w) &= [S(\beta(2,w)\%2)*\beta(1,w)]*\lfloor \tfrac{1}{2}\beta(2,w)\rfloor*0 \\ p_1(n)&=n \\ q_1(n)&=4n+6. \end{aligned}$$

Then define $Encode(x)=\beta(1,f_1(x))$ and $Init(x)=0*0*Encode(x)*0*0*0$.

We define *Decode* to be the inverse of *Encode* as follows: define f_2 by limited iteration from g_2 and h_2 with bounds p_2 and q_2, where

$$\begin{aligned} g_2(x) &= x*0*0 = <x,0> \\ k_2(w) &= Truncate(\beta(1,w))*Choice(\beta(1,\beta(1,w))\equiv 1,2\cdot\beta(2,w),S(2\cdot\beta(2,w)))*0 \\ h_2(x,m,w) &= Choice(\beta(1,w),k_2(w),w) \\ p_2(n) &= n \\ q_2(n) &= 2n+4. \end{aligned}$$

Then define $Decode(x)=\beta(2,f_2(x))$.

Next is the function which maps the Gödel number of an ID of M to the Gödel number of the next ID of M. We sketch how *Next* is defined using composition only (no further use of limited iteration). First note that t_0, s_0 and u are given by

$$\begin{aligned} T_0(x) &= Choice(\beta(3,x),\beta(1,\beta(3,x)),0) \\ S_0(x) &= Choice(\beta(5,x),\beta(1,\beta(5,x)),0) \\ U(x) &= \beta(1,x). \end{aligned}$$

The oracle queries are given by (for $i=1,2,\ldots,k$):

$$H_i(x) = Encode(h_i(Decode(\beta(5,x)))).$$

It should now be clear that *Next* can be defined by the use of many *Choice* functions and simple composition from the above functions and the functions in B^+.

We finally define f_3 by limited iteration from *Init* and *Next* with time bound p and space bound q_3. Recall p is the bound on the runtime of M. q_3 is the polynomial $q_3(n) = 8\cdot(|N+k+J+2|)\cdot(q(n)+1)$. So q_3 bounds the length of the Gödel numbers of ID's of M. Now define

$$f(x) = Decode(\beta(3,f_3(x))),$$

and f is the function M computes and by construction f is in $PTC(C)$.

Q.E.D. □

1.3. Bounded Quantifiers.

Quantification is a construction which forms an n–ary predicate from an $(n+1)$–ary predicate. For this chapter only we adopt the convention that a *predicate* is a function with range $\{0,1\}$ where 0 denotes *False* and 1 denotes *True.*

Definition: Let C be a set of functions. Then $PRED(C)$ is the set of predicates in C, i.e., the functions in C with range $\{0,1\}$.

Definition: Let Q and R be functions. Then $(\forall y \leq Q(\vec{x}))R(\vec{x},y)$ is the predicate (i.e., function of $\vec{x}$) which has value 1 iff for all $y \leq Q(\vec{x})$ the value of $R(\vec{x},y)$ is nonzero. Similarly, $(\exists y \leq Q(\vec{x}))R(\vec{x},y)$ is the predicate which has value 1 iff for some $y \leq Q(\vec{x})$ the value of $R(\vec{x},y)$ is nonzero. (Note that this definition applies even if R is not a predicate.)

We will be interested only in bounded quantification, that is to say, in quantifiers of the form $(\forall x \leq t)$ or $(\exists x \leq t)$. Indeed, if we used unbounded quantification the construction below would just give the arithmetic hierarchy since the class Δ_0^p defined below includes a version of the Kleene T predicate.

We define two kinds of bounded quantification which are distinguished by the size of the bound. *Polynomially Bounded Quantification* allows bounds of the form $2^{p(|t|)}$, where p is a polynomial; whereas *Logarithmically Bounded Quantification* allows only bounds of the form $p(|t|)$.

Definition: Let C be a set of functions closed under composition. Then $PB\exists(C)$ is the set of predicates Q such that

(1) $Q:\mathbf{N}^i \rightarrow \mathbf{N}$ for some $i \in \mathbf{N}$;
(2) There is an $R \in PRED(C)$ and a suitable polynomial p such that for all $\vec{x}$,

$$Q(\vec{x}) = (\exists y \leq 2^{p(|\vec{x}|)})R(\vec{x},y).$$

$PB\forall$ is defined similarly with a universal quantifier replacing the existential quantifier in (2). Note that $PB\forall(C)$ and $PB\exists(C)$ always contain $PRED(C)$.

Definition: Let C be a set of functions closed under composition. Then $LB\exists(C)$ is the set of predicates Q such that

(1) $Q:\mathbf{N}^i \rightarrow \mathbf{N}$ for some $i \in \mathbf{N}$;
(2) There is an $R \in PRED(C)$ and a suitable polynomial p such that for all $\vec{x}$,

$$Q(\vec{x}) = (\exists y \leq p(|\vec{x}|))R(\vec{x},y).$$

$LB\forall$ is defined similarly with a universal quantifier replacing the existential quantifier in (2). Note that $LB\forall(C)$ and $LB\exists(C)$ always contain $PRED(C)$.

In later chapters we will define bounded quantification in a different setting. Logarithmically bounded quantification corresponds to what we later call *sharply bounded quantification*. Our definition of logarithmically bounded quantification is closely related to what Bennet [3] called "part of" quantification and polynomially bounded quantification corresponds to what he called "finite" quantification.

1.4. The Polynomial Hierarchy.

We are now in a position to define the polynomial hierarchy. We will differ from the usual definitions in that we define a hierarchy of functions as well as a hierarchy of predicates.

Definition: (by induction on k)

(1) $\Box_0^p$ is the smallest set of functions containing B and closed under composition, $LB\exists$ and $LB\forall$.

(2) $\Delta_0^p = \Sigma_0^p = \Pi_0^p = PRED(\Box_0^p)$.

(3) $\Box_{k+1}^p = PTC(\Sigma_k^p)$.

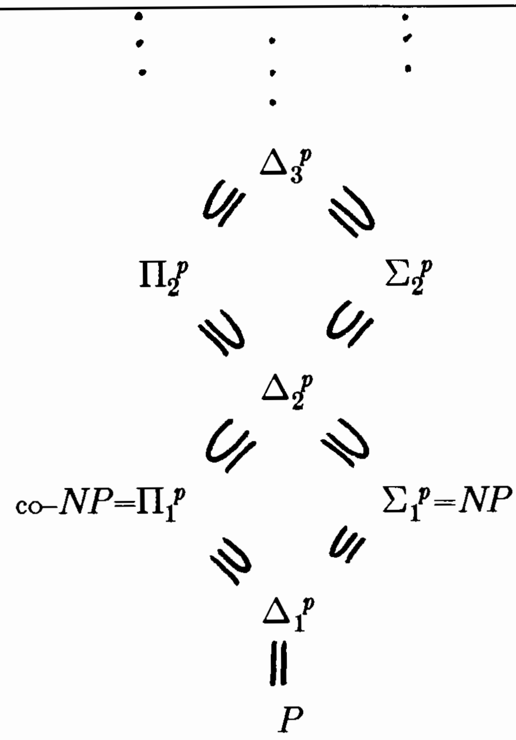
(4) $\Delta_{k+1}^p = PRED(\Box_{k+1}^p)$.

(5) $\Sigma_{k+1}^p = PB\exists(\Delta_{k+1}^p)$.

(6) $\Pi_{k+1}^p = PB\forall(\Delta_{k+1}^p)$.

(7) $PH = \bigcup_k \Sigma_k^p$.

The sets of predicates Δ_1^p, Σ_1^p and Π_1^p are well known to computer scientists and are called P, NP and co-NP respectively. Figure 1 shows a diagram of the hierarchy of predicates Δ_k^p, Σ_k^p and Π_k^p.



The Polynomial Hierarchy
Figure 1

Proposition 4: $\square_{k+1}^p = PTC(\Pi_k^p)$ for all $k \geq 0$.

Proof: This is easy and is left as an exercise for the reader. □

There are many open problems concerning the polynomial hierarchy. We say the hierarchy *collapses* if there is a k such that $\Sigma_k^p = \Sigma_{k+1}^p$. Otherwise we say that the hierarchy is *proper*. Things which we do **not** know include:

(1) Does $P = NP$?
(2) Does $NP =$ co-NP?
(3) Does the polynomial hierarchy collapse?
(4) Does $\Delta_k^p = \Sigma_k^p \cap \Pi_k^p$? In particular, does $P = NP \cap$ co-NP?

Most computer scientists are of the opinion that all these questions have negative answers, especially the first two. However, over a decade of determined efforts has failed to resolve these questions.

One question we can answer is whether $\Delta_0^p = \Delta_1^p$:

Proposition 5: $\Delta_0^p \neq \Delta_1^p$.

Proof: Let $Parity : \mathbf{N} \longrightarrow \mathbf{N}$ be the function defined as

$$\begin{aligned} Numones(x) &= \#\text{ of ones in the binary representation of } x \\ Parity(x) &= Numones(x)\%2. \end{aligned}$$

Clearly, $Parity \in \Delta_1^p = P$. So it suffices to show that $Parity \notin \Delta_0^p$.

It is easy to show that if $f \in \Delta_0^p$ then f has polynomial size, unbounded fan-in circuits of constant depth. This is proved by induction on the complexity of the definition of f: the only two cases are composition and logarithmically bounded quantification and both are straightforward. But Furst, Saxe and Sipser [11] have shown that *Parity* does not have constant depth, polynomial size circuits. □

Proposition 5 is somewhat unsatisfactory as it depends on the fact that the initial functions in B all have constant depth polynomial size circuits. Indeed if multiplication had been included in B it would no longer be true that all functions in B have constant depth polynomial size circuits. It would be desirable to establish a more general version of Proposition 5 (if, in fact, a more general version is true.)

1.5. Eliminating PTC.

In defining the polynomial hierarchy we alternately applied PTC (polynomial time closure) and $PB\exists$ (polynomially bounded quantification). It turns out that the use of PTC is unnecessary and that the classes Σ_k^p and Π_k^p can be defined without using PTC and hence without using either Turing machines or limited iteration.

Lemma 6:

(a) For all $k \geq 0$, Δ_k^p is closed under logarithmically bounded quantification ($LB\forall$ and $LB\exists$), conjunction, disjunction and negation.

(b) For all $k \geq 0$, Π_k^p and Σ_k^p are closed under $LB\exists$, $LB\forall$, conjunction and disjunction.

Proof:

(a) This is immediate from the definition of Δ_k^p except for showing closure under $LB\forall$ and $LB\exists$ when $k \geq 1$. Suppose that $R \in \Delta_k^p$ and Q is defined by

$$Q(\vec{x}) = (\forall x \leq p(|\vec{x}|))R(x,z).$$

We can define $Q(\vec{x})$ by limited iteration from g and h with bounds p and q, where

$$g(\vec{x}) = R(\vec{x},0)$$
$$h(\vec{x},m,y) = y\wedge R(x,Sm)$$
$$q(n) = 1.$$

Since g and h are in Δ_k^p, so is Q. This shows Δ_k^p is closed under $LB\forall$ and a similar argument shows it is closed under $LB\exists$.

(b) Since $\Sigma_0^b = \Pi_0^b = \Delta_0^b$, (b) is just a special case of (a) when k=0. So suppose $k\geq 1$. The closure of Π_k^p and Σ_k^p under conjunction and disjunction follows easily from (a). To show that Σ_k^p is closed under $LB\forall$ it suffices to show that if $R\in\Delta_{k-1}^p$ and if p and q are suitable polynomials, then

$$S(\vec{x}) = (\forall z\leq p(|\vec{x}|))(\exists y\leq 2^{q(|\vec{x}|,z)})R(\vec{x},y,z)$$

is in Σ_k^p. But $S(\vec{x})$ is equivalent to

$$(\exists y\leq 2^{r(|\vec{x}|)})(\forall z\leq p(|\vec{x}|))\,[R(\vec{x},\beta(Sz,y),z)\wedge\beta(Sz,y)\leq 2^{q(|\vec{x}|,z)}]$$

where $r(\vec{n}) = 2\cdot(q(\vec{n},p(\vec{n}))+2)\cdot(p(\vec{n})+1)$. Thus $S(\vec{x})$ is in Σ_k^p. A similar argument shows Π_k^p is closed under $LB\exists$.

Q.E.D. □

The next theorem shows how PTC can be eliminated from the definition of the polynomial hierarchy.

Theorem 7: (Meyer-Stockmeyer-Wrathall).

(a) For all $k\geq 1$, $\Sigma_{k+1}^p = PB\exists(\Pi_k^p)$ and $\Pi_{k+1}^p = PB\forall(\Sigma_k^p)$.

(b) Let B^* be the smallest set containing B^+ which is closed under $LB\forall$, $LB\exists$, and composition. Then $\Sigma_1^b{=}PB\exists(B^*)$ and $\Pi_1^b{=}PB\forall(B^*)$.

Proof: In order to prove (a) and (b) simultaneously, we define D_{k+1} to be Π_{k+1}^p and E_{k+1} to be Σ_{k+1}^p, and $D_0 = E_0$ to be B^*. The theorem asserts that $\Sigma_{k+1}^p = PB\exists(D_k)$ and $\Pi_{k+1}^p = PB\forall(E_k)$ for all $k\geq 0$. It suffices to show that $\Sigma_{k+1}^b = PB\exists(D_k)$ since $\Pi_{k+1}^b = PB\forall(E_k)$ is an immediate consequence of this.

Let k be a fixed nonnegative integer. Directly from the definitions we have $\Sigma_{k+1}^p\supseteq PB\exists(D_k)$. We need to show the reverse inclusion also holds. Let C_0 be D_k. Define C_i^* to be the set of functions definable by a single use of limited iteration from functions in C_i. Set C_{i+1} equal to the closure of C_i^* under composition.

We will show that for all i, $PB\exists(D_k)\supseteq PB\exists(C_i)$. Since $\cup C_i{=}\square_{k+1}^p$, this suffices to prove the theorem. We will show by induction on i that for any $Q\in C_i$,

$$S(\vec{x}) \;=\; (\exists z \le 2^{p(|\vec{x}|)})(\forall y \le q(|\vec{x}|))Q(\vec{x},y,z)$$

is in $PB\exists(D_k)$. (This may seem like an unusual definition for S but it makes the induction argument work well.) This is easily seen to be true when i=0 since C_0 is D_k and by Lemma 6(b) D_k is closed under $PB\forall$. So assume $i>0$. Without loss of generality we may assume Q has the form

$$Q(\vec{s}) \;=\; G(\vec{s},F_1(\vec{s}),\ldots,F_n(\vec{s})).$$

where G is in D_k and each F_i is in C^*_{i-1}. (If this is not the case we can find a formula equivalent to Q in this form. For example, $Q(\vec{s})=G(F_1(F_2(\vec{s})))$ is equivalent to the formula $(\exists v \le 2^{q(|\vec{s}|)})(v \equiv F_2(|\vec{s}|) \wedge G(F_1(v)))$, where q is a suitable polynomial which bounds the function F_2. The extra existential quantifier introduced by this may be eliminated from S by first interchanging it with the logarithmically bounded quantifier in S by using the trick of the proof of Lemma 6(b), and then combining it with the original existential quantifier of S by using the pairing function. Note that the β function is always in D_k and hence it is permissible for G to involve the pairing function.)

Let each F_j be defined by limited iteration from G_j and H_j with time bound p_j and space bound q_j, where G_j and H_j are in C_{i-1}.

We informally define *ValidComp* $(w,\vec{s})$ to be *True* iff

(1) w is a sequence $<w_1,\ldots,w_n>$ and
(2) Each w_j codes a sequence $<w_{j,0},\ldots,w_{j,n_j}>$ which codes the computation of $F_j(\vec{s})$.

A precise definition is:

$$\begin{aligned} \mathit{ValidComp}\,(w,\vec{s}) \;=\; & \beta(0,w) \equiv n \wedge \bigwedge_{j=1}^{k}(w_{j,1} \equiv G_j(\vec{s})) \wedge \\ & \wedge \bigwedge_{j=1}^{k}(\beta(0,w_j) \equiv p_j(|\vec{s}|)+1) \wedge \\ & \wedge \bigwedge_{j=1}^{k}(\forall v \le |w|)[Sv < \beta(0,w_j) \supset w_{j,v+2} \equiv H_j(\vec{s},v,w_{j,v+1})] \end{aligned}$$

where we used the abbreviations w_j for $\beta(j,w)$ and $w_{j,m}$ for $\beta(m,\beta(j,w))$.

Now we can easily find a suitable polynomial r large enough so that $Q(\vec{s})$ is equivalent to

$$(\exists w \le 2^{r(|\vec{s}|)})[\mathit{ValidComp}\,(w,\vec{s}) \wedge G(\vec{s},w_{1,\beta(0,w_1)},\ldots,w_{n,\beta(0,w_n)})].$$

The only quantifiers in *ValidComp* are logarithmically bounded quantifiers, so we may rewrite this last equation as

$$(\exists w \leq 2^{r(|\vec{s}|)})(\forall v \leq |w|)R(\vec{s},v,w)$$

where $R \in C_{i-1}$. So $S(\vec{x})$ is equivalent to

$$(\exists z \leq 2^{p(|\vec{x}|)})(\forall y \leq q(|\vec{x}|))(\exists w \leq 2^{r(|\vec{x}|,|y|,|z|)})(\forall v \leq |w|)R(\vec{x},y,z,v,w).$$

Now we can use the method of the proof of Lemma 6(b) to interchange the order of the second and third quantifiers. We then can use the β function as a pairing function to contract adjacent like quantifiers (since the β function is in D_k). Hence $S(\vec{x})$ is equivalent to

$$(\exists z^* \leq 2^{s(|\vec{x}|)})(\forall y^* \leq t(|\vec{x}|))R^*(\vec{x},z^*,y^*)$$

where s and t are suitable polynomials and $R^* \in C_{i-1}$. By the induction hypothesis, $S(\vec{x})$ is in $PB\exists(D_k)$, which completes the induction step and the proof.

Q.E.D. □

The point of Theorem 7 is that we now can characterize the classes Σ_k^p and Π_k^p of the polynomial hierarchy in a purely syntactic way. We start with the initial set B^+ of functions and take its closure under composition and logarithmically bounded quantification to obtain B^*. We apply polynomially bounded quantification repeatedly to obtain Σ_k^p and Π_k^p. (A somewhat stronger result is obtained by Kent-Hodgson [17].)

Hence the question of whether the polynomial hierarchy collapses is the question of whether there is a "quantifier elimination" theorem for polynomially bounded quantifiers.

1.6. Bounded Arithmetic Formulae.

An *arithmetic formula* is a formula of first order logic which may contain the logical symbols $\wedge$, $\vee$, $\neg$, $\exists$, $\forall$, $\supset$, and $=$ and the non-logical symbols 0, S, $+$, $\cdot$, $\#$, $|x|$, $\lfloor \frac{1}{2}x \rfloor$, and $\leq$. The non-logical symbols have the following meanings:

0	zero constant symbol
S	successor
$+$	addition
$\cdot$	multiplication
$\lfloor \frac{1}{2}x \rfloor$	"shift right" function
$\|x\|$	$= \lceil \log_2(x+1) \rceil$, the length of the binary representation of x

$x\#y$	$= 2^{|x|\cdot|y|}$, the "smash" function
$\leq$	less than or equal to

A *bounded quantifier* is one of the form $(\forall x \leq t)$ or $(\exists x \leq t)$ where t can be any term. A *sharply bounded quantifier* is one of the form $(\forall x \leq |t|)$ or $(\exists x \leq |t|)$. Note that if p is any suitable polynomial then the $\#$, $\cdot$, and $\lfloor \frac{1}{2}x \rfloor$ functions can be used to form a term equal to $2^{p(|\vec{x}|)}$. Thus bounded and sharply bounded quantifiers correspond precisely to the polynomially and logarithmically bounded quantifiers, respectively.

An *unbounded quantifier* is a regular quantifier of the form $(\forall x)$ or $(\exists x)$. An arithmetic formula is *bounded* iff it contains no unbounded quantifiers.

We define a hierarchy of bounded arithmetic formulae as follows:

Definition: The following sets of formulae are defined by induction on the complexity of formulae:

(1) $\Pi_0^b = \Sigma_0^b = \Delta_0^b$ is the set of formulae all of whose quantifiers are sharply bounded.

(2) Σ_{k+1}^b is defined inductively by:
- (a) $\Sigma_{k+1}^b \supseteq \Pi_k^b$
- (b) If A is in Σ_{k+1}^b then so are $(\exists x \leq t)A$ and $(\forall x \leq |t|)A$.
- (c) If $A, B \in \Sigma_{k+1}^b$ then $A \wedge B$ and $A \vee B$ are in Σ_{k+1}^b.
- (d) If $A \in \Sigma_{k+1}^b$ and $B \in \Pi_{k+1}^b$ then $\neg B$ and $B \supset A$ are in Σ_{k+1}^b.

(3) Π_{k+1}^b is defined inductively by:
- (a) $\Pi_{k+1}^b \supseteq \Sigma_k^b$
- (b) If A is in Π_{k+1}^b then so are $(\forall x \leq t)A$ and $(\exists x \leq |t|)A$.
- (c) If $A, B \in \Pi_{k+1}^b$ then $A \wedge B$ and $A \vee B$ are in Π_{k+1}^b.
- (d) If $A \in \Pi_{k+1}^b$ and $B \in \Sigma_{k+1}^b$ then $\neg B$ and $B \supset A$ are in Π_{k+1}^b.

(4) Σ_{k+1}^b and Π_{k+1}^b are the smallest sets which satisfy (1)-(3).

This hierarchy of bounded formulae is in many respects analogous to the arithmetic hierarchy. The classes Σ_k^b and Π_k^b are defined by counting alternations of bounded quantifiers, ignoring the sharply bounded quantifiers. The arithmetic hierarchy is defined by counting alternations of unbounded quantifiers, ignoring the bounded quantifiers. We are using bounded and sharply bounded quantifiers in a manner analogous to the use of unbounded and bounded quantifiers (respectively) in the arithmetic hierarchy.

Theorem 8: Let $k \geq 1$. Σ_k^p (respectively, Π_k^p) is the class of predicates which are defined by formulae in Σ_k^b (respectively, Π_k^b).

Proof: By Theorem 7, Lemma 6, and the definition of the bounded arithmetic hierarchy, it suffices to prove the theorem for the case $k=1$.

First we show Σ_1^p contains all predicates defined by Σ_1^b formulae. All the nonlogical symbols of bounded arithmetic can be computed in polynomial time and hence are in $\square_1^p$. Since $\square_1^p$ is closed under composition and since Σ_1^p is closed under conjunction, disjunction and logarithmically bounded quantification, the desired result is established. The same argument also shows that Π_1^p contains all predicates defined by Π_1^b predicates.

For the reverse inclusion, let R be an arbitrary predicate in Σ_1^p. By Theorem 7, R can be written in the form

$$R(\vec{x}) = (\exists y \le 2^{p(|\vec{x}|)}) S(\vec{x}, y)$$

with $S \in D_0$, where, as in the proof of Theorem 7, D_0 is the smallest set of functions containing B^+ and closed under composition and logarithmically bounded quantification. In other words, S is expressible by a formula which uses functions from B^+ and logarithmically bounded quantification.

So to show R is definable by a Σ_1^b–formula, it will suffice to show that S is definable by a Σ_1^b–formula. To show that, we have to show that every occurrence of *Choice*, *Truncate*, $*$ and β can be replaced by an equivalent arithmetic formula.

The simplest case is eliminating *Choice* from S. Suppose S is $F(Choice(a,b,c))$. Then S is equivalent to

$$(F(b) \wedge \neg a=0) \vee (F(c) \wedge a=0).$$

By repeated transformations of this type, all occurrences of *Choice* can be eliminated from S.

Eliminating *Truncate*, β, and $*$ is a little more difficult. We shall show in great detail in Chapter 2 that S is in fact equivalent to a Σ_1^b–formula. In particular, see Theorem 2.2 in §2.3 and also see §2.4 and §2.5. So we omit the proof here.

Since the Π_k^p predicates are the negations of the Σ_k^p predicates and the Π_k^b–formulae are equivalent to the negations of the Σ_k^b–formulae, we have immediately from the above that the Π_k^p predicates are precisely the predicates definable by Π_k^b–formulae (when $k \ge 1$).

Q.E.D. □

1.7. Relativization of the Polynomial Hierarchy.

The polynomial hierarchy can be relativized by allowing Turing machines to query oracles. Recall that we already defined in §1.2 what it means for a Turing machine to use a function oracle.

Definition: A *function oracle* Ω is a function of polynomial growth rate whose domain is $\mathbf{N}^k$ for some $k \geq 1$ and range is $\mathbf{N}$. A *predicate oracle* is a function oracle which has range $\{0,1\}$.

Definition: Let $\Omega_1, \ldots, \Omega_k$ be a sequence of function oracles. The following classes of functions and predicates are defined inductively on i:

$$
\begin{aligned}
&(1)\ \square_1^p(\Omega_1, \ldots, \Omega_k) &=&\ PTC(\Omega_1, \ldots, \Omega_k) \\
&(2)\ \Delta_{i+1}^p(\Omega_1, \ldots, \Omega_k) &=&\ PRED(\square_{i+1}^p(\Omega_1, \ldots, \Omega_k)) \\
&(3)\ \Sigma_{i+1}^p(\Omega_1, \ldots, \Omega_k) &=&\ PB\exists(\Delta_{i+1}^p(\Omega_1, \ldots, \Omega_k)) \\
&(4)\ \Pi_{i+1}^p(\Omega_1, \ldots, \Omega_k) &=&\ PB\forall(\Delta_{i+1}^p(\Omega_1, \ldots, \Omega_k)) \\
&(5)\ \square_{i+2}^p(\Omega_1, \ldots, \Omega_k) &=&\ PTC(\Sigma_{i+1}^p(\Omega_1, \ldots, \Omega_k)) \\
&(6)\ PH(\Omega_1, \ldots, \Omega_k) &=&\ \bigcup_i \Sigma_i^p(\Omega_1, \ldots, \Omega_k)
\end{aligned}
$$

The definition above gives us a relativization of the polynomial hierarchy for each fixed sequence of oracles $\Omega_1, \ldots, \Omega_k$. We shall also need a more general concept of relativizing with respect to an *arbitrary* set of oracles. We do this by the definitions below.

Definition: Let j be a positive integer and let $p(x_1, \ldots, x_j)$ be a suitable polynomial. Then ω_j^p is equal to the set of all j-ary function oracles Ω satisfying $|\Omega(\vec{n})| \leq p(|\vec{n}|)$ for all $\vec{n} \in \mathbf{N}^j$.

Definition: A *functional* f is a function with domain

$$\mathbf{N}^{k_0} \times \omega_{k_1}^{p_1} \times \cdots \times \omega_{k_i}^{p_i}$$

and range $\mathbf{N}$ where $i \geq 0$ and each $k_j \geq 1$ and each p_j is a suitable polynomial. Thus a functional maps a tuple of k_0 integers and i function oracles to a nonnegative integer. Such a functional is called k_0-ary.

The functional f has *polynomial growth rate* iff there is a suitable polynomial $r(\vec{x})$ such that for all $\vec{n} \in \mathbf{N}^{k_0}$ and all function oracles $\Omega_1, \ldots, \Omega_i$ with $\Omega_j \in \omega_{k_j}^{p_j}$ for $1 \leq j \leq i$ we have

$$|f(\vec{n}, \vec{\Omega})| \leq r(|\vec{n}|).$$

We next need to relativize the definitions of PTC, $PRED$, $PB\forall$ and $PB\exists$.

Definition: Let C be a set of functionals. Then $PRED(C)$ is the set of members of C which have range $\{0,1\}$.

Definition: Let g and h be functionals such that the domain of g is

$$\mathbf{N}^k \times \omega_{n_1}^{p_1} \times \cdots \times \omega_{n_i}^{p_i}$$

and the domain of h is

$$\mathbf{N}^{k+2} \times \omega_{n_1}^{p_1} \times \cdots \times \omega_{n_i}^{p_i}.$$

Let p and q be k-ary suitable polynomials. Then f is *defined by limited iteration from g and h with time bound p and space bound q* iff the following holds:

Let τ be the functional with domain

$$\mathbf{N}^{k+1} \times \omega_{n_1}^{p_1} \times \cdots \times \omega_{n_i}^{p_i}$$

so that for all oracles $\Omega_1, \ldots, \Omega_i$ with $\Omega_j \in \omega_{n_j}^{p_j}$ for $1 \leq j \leq i$ and for all $\vec{x} \in \mathbf{N}^k$, τ is defined by

$$\tau(x_1, \ldots, x_k, 0, \Omega_1, \ldots, \Omega_i) = g(x_1, \ldots, x_k, \Omega_1, \ldots, \Omega_i)$$

$$\tau(x_1, \ldots, x_k, n+1, \Omega_1, \ldots, \Omega_i) = h(x_1, \ldots, x_k, n, \tau(x_1, \ldots, x_k, n, \Omega_1, \ldots, \Omega_i), \Omega_1, \ldots, \Omega_i).$$

And we must have that for all $\vec{x}$, n and $\vec{\Omega}$ as above

$$|\tau(\vec{x}, n, \vec{\Omega})| \leq q(|\vec{x}|)$$

and

$$f(x_1, \ldots, x_k, n, \Omega_1, \ldots, \Omega_i) = \tau(x_1, \ldots, x_k, p(|\vec{x}|), \Omega_1, \ldots, \Omega_i).$$

Definition: Let C be a set of functionals. We say that C is *uniform* iff there exists $\omega_{n_1}^{p_1}, \ldots, \omega_{n_i}^{p_i}$ such that every functional $f \in C$ has domain

$$\mathbf{N}^{k_f} \times \omega_{n_1}^{p_1} \times \cdots \times \omega_{n_i}^{p_i}$$

for some k_f which depends on f.

Definition: Let C be a uniform set of functionals of polynomial growth rate. The domain of each functional in C is of the form

$$\mathbf{N}^k \times \omega_{n_1}^{p_1} \times \cdots \times \omega_{n_i}^{p_i}$$

for some fixed $\omega_{n_1}^{p_1}, \ldots, \omega_{n_i}^{p_i}$. The *Polynomial-time closure*, $PTC(C)$, of C is the smallest uniform set of functionals containing C such the following hold:

(1) For each n-ary function $f \in B$, there is an n-ary functional $g \in PTC(C)$ such that for all $\vec{x}$ and all $\vec{\Omega}$,

$$g(\vec{x}, \vec{\Omega}) = f(\vec{x}).$$

(2) For each $1 \leq j \leq i$, the functional P_j defined by

$$P_j(x_1, \ldots, x_{n_j}, \Omega_1, \ldots, \Omega_i) = \Omega_j(x_1, \ldots, x_{n_j})$$

is in C.

(3) C is closed under composition and under definition by limited iteration.

Definition: Let C be a set of functionals. Then $PB\exists(C)$ is the set of functionals Q such that Q has range $\{0,1\}$ and domain

$$\mathbf{N}^k \times \omega_{n_1}^{p_1} \times \cdots \times \omega_{n_i}^{p_i}$$

and such that there exists a suitable polynomial p and an $R \in PRED(C)$ with domain

$$\mathbf{N}^{k+1} \times \omega_{n_1}^{p_1} \times \cdots \times \omega_{n_i}^{p_i}$$

such that for all $\Omega_1, \ldots, \Omega_i$ with $\Omega_j \in \omega_{n_j}^{p_j}$ for $1 \leq j \leq i$, we have

$$Q(x_1, \ldots, x_k, \Omega_1, \ldots, \Omega_i) = (\exists y \leq 2^{p(|\vec{x}|)}) R(x_1, \ldots, x_k, y, \Omega_1, \ldots, \Omega_i).$$

$PB\forall(C)$ is defined similarly except that a bounded universal quantifier $(\forall y \leq 2^{p(|\vec{x}|)})$ is used instead of the bounded existential quantifier.

We next define a polynomial hierarchy of functionals:

Definition: Let $\omega_{n_1}^{p_1}, \ldots, \omega_{n_i}^{p_i}$ be a sequence of function oracles. The classes defined below are uniform sets of functionals which have domains of the form

$$\mathbf{N}^k \times \omega_{n_1}^{p_1} \times \cdots \times \omega_{n_i}^{p_i}.$$

The definition is by induction on j:

(1) $\Box_1^p(\omega_{n_1}^{p_1}, \ldots, \omega_{n_i}^{p_i}) = PTC(\emptyset)$

(2) $\Delta_{j+1}^p(\omega_{n_1}^{p_1}, \ldots, \omega_{n_i}^{p_i}) = PRED(\Box_{j+1}^p(\omega_{n_1}^{p_1}, \ldots, \omega_{n_i}^{p_i}))$

(3) $\Sigma_{j+1}^p(\omega_{n_1}^{p_1}, \ldots, \omega_{n_i}^{p_i}) = PB\exists(\Delta_{j+1}^p(\omega_{n_1}^{p_1}, \ldots, \omega_{n_i}^{p_i}))$

(4) $\Pi_{j+1}^p(\omega_{n_1}^{p_1}, \ldots, \omega_{n_i}^{p_i}) = PB\forall(\Delta_{j+1}^p(\omega_{n_1}^{p_1}, \ldots, \omega_{n_i}^{p_i}))$

(5) $\Box_{j+2}^p(\omega_{n_1}^{p_1}, \ldots, \omega_{n_i}^{p_i}) = PTC(\Sigma_{j+1}^p(\omega_{n_1}^{p_1}, \ldots, \omega_{n_i}^{p_i}))$

(6) $PH(\omega_{n_1}^{p_1}, \ldots, \omega_{n_i}^{p_i}) \ = \ \bigcup_j \Sigma_j^p(\omega_{n_1}^{p_1}, \ldots, \omega_{n_i}^{p_i})$

Proposition 10: Let $\omega_{n_1}^{p_1}, \ldots, \omega_{n_i}^{p_i}$ be a sequence of function oracles and let $\Omega_1, \ldots, \Omega_i$ be oracles so that $\Omega_j \in \omega_{n_j}^{p_j}$ for all $1 \le j \le i$. Let $k \ge 1$. Then for all functions f, $f \in \Box_k^p(\Omega_1, \ldots, \Omega_i)$ iff there exists a functional $g \in \Box_k^p(\omega_{n_1}^{p_1}, \ldots, \omega_{n_i}^{p_i})$ such that for all $\vec{x}$,

$$f(\vec{x}) \ = \ g(\vec{x}, \Omega_1, \ldots, \Omega_i).$$

Similar statements hold for $\Delta_k^p(\Omega_1, \ldots, \Omega_i)$, $\Sigma_k^p(\Omega_1, \ldots, \Omega_i)$ and $\Pi_k^p(\Omega_1, \ldots, \Omega_i)$.

The proof of Proposition 10 is not too difficult and we omit it.

1.8. Appendix.

We prove Theorem 1 in this appendix.

Theorem 1: $PTC(\emptyset) \supseteq B^+$.

Proof: We define the functions of B^+ by limited iteration from functions in B.

(1) Define $Bit: \mathbf{N}^2 \to \mathbf{N}$ by limited iteration from g_1 and h_1 with bounds p_1 and q_1, where

$$\begin{aligned} g_1(i,x) &= x \\ h_1(i,x,m,v) &= Choice(m<i, \lfloor \tfrac{1}{2} v \rfloor, v\%2) \\ p_1(n,m) &= m \\ q_1(n,m) &= m \end{aligned}$$

So if the binary representation of x is $x_{m-1} \cdots x_0$ then $Bit(i,x)$ is equal to x_i.

(2) Define $f_2:\mathbf{N}^2 \rightarrow \mathbf{N}$ by limited iteration from g_2 and h_2 with bounds p_2 and q_2, where

$$\begin{aligned} g_2(a,w) &= 2\cdot 2\cdot w \\ h_2(a,w,m,v) &= Choice(Bit(m,a),S(2(S(2v))),2(S(2v))) \\ p_2(n,m) &= n \\ q_2(n,m) &= m+2n+2 \end{aligned}$$

Set $a*w = Choice(a=0,2(S(2\cdot 2\cdot 2\cdot w)),f_2(a,w))$.

(3) Define $f_3:\mathbf{N}\rightarrow\mathbf{N}$ by limited iteration from g_3 and h_3 with bounds p_3 and q_3, where

$$\begin{aligned} g_3(w) &= w \\ h_3(w,m,v) &= Choice(4\cdot\lfloor v/4\rfloor = v, v, \lfloor v/4\rfloor) \\ p_3(m) &= m \\ q_3(m) &= m \end{aligned}$$

Set $Truncate(w) = \lfloor f_3(w)/4\rfloor$.

(4) Define $TR\,(i,w):\mathbf{N}^2\rightarrow\mathbf{N}$ by limited iteration from g_4 and h_4 with bounds p_4 and q_4, where

$$\begin{aligned} g_4(i,w) &= w \\ h_4(i,w,m,v) &= Choice(Sm<i, Truncate(v), v) \\ p_4(n,m) &= m \\ q_4(n,m) &= m \end{aligned}$$

So $TR\,(i,w)$ is $Truncate$ applied $i \dot{-} 1$ times to w.

(5) Define $f_5:\mathbf{N}^2\rightarrow\mathbf{N}$ by limited iteration from g_5 and h_5 with bounds p_5 and q_5, where

$$\begin{aligned} g_5(i,w) &= 0 \\ h_5(i,w,m,v) &= v\vee(\neg Bit(2m,w)\wedge\neg Bit(S(2m),w)\wedge m<i) \\ p_5(n,m) &= m \\ q_5(n,m) &= 1 \end{aligned}$$

$$\text{So } f_5(i,w) = \begin{cases} 1 & \text{if } |\beta(1,w)|<i \\ 0 & \text{otherwise} \end{cases}$$

(6) Define $f_6:\mathbf{N}\rightarrow\mathbf{N}$ by limited iteration from g_6 and h_6 with bounds p_6 and q_6, where

$$\begin{aligned}
g_6(w) &= 0\\
h_6(w,m,v) &= Choice(f_5(m,w),Choice(Bit(2m,w),S(2v),2v),v)\\
p_6(n) &= n\\
q_6(n) &= n
\end{aligned}$$

So $f_6(w) = \beta(1,w)$, the value of the first element in the sequence w.

(7) Define $f_7:\mathbf{N}\rightarrow\mathbf{N}$ by limited iteration from g_7 and h_7 with bounds p_7 and q_7, where

$$\begin{aligned}
g_7(w) &= 0\\
h_7(w,m,v) &= Choice(\mathit{TR}\ (m,w)\wedge\neg \mathit{TR}\ (Sm,w),m,v)\\
p_7(n) &= n\\
q_7(n) &= n
\end{aligned}$$

So $f_7(w) = \beta(0,w)$, the number of elements in the sequence w.

(8) Define $\beta:\mathbf{N}^2\rightarrow\mathbf{N}$ by

$$\beta(i,w) = Choice(i,f_6(\mathit{TR}\ (i,w)),f_7(w)).$$

Q.E.D. □

Chapter 2

Foundations of Bounded Arithmetic

Bounded Arithmetic is a weak fragment of Peano arithmetic and is of interest to us because of its connections to the polynomial hierarchy. It will take us a fair amount of work to establish the relationship between Bounded Arithmetic and the polynomial hierarchy. This chapter is devoted to establishing the foundations of Bounded Arithmetic; in particular, we define some useful axiomatizations of fragments of Bounded Arithmetic.

2.1. The Language of Bounded Arithmetic.

The first order language of Bounded Arithmetic contains all the usual logical symbols $\wedge$, $\vee$, $\neg$, $\supset$, $=$, $\exists$, $\forall$ and parentheses and the nonlogical function symbols S, 0, $+$, $\cdot$, $|x|$, $\lfloor\frac{1}{2}x\rfloor$, and $\#$ and the nonlogical predicate symbol $\leq$. These nonlogical symbols are intended to be applied to nonnegative integers; from now on, we use "integer" or "number" to mean nonnegative integer. S, 0, $+$, $\cdot$, and $\leq$ are the successor function, the zero constant, addition, multiplication, and the less-than-or-equal-to relation. $|x|$ denotes the length of the binary representation of x; i.e. $|x| = \lceil\log_2(x+1)\rceil$. For example, $|0| = 0$. $\lfloor\frac{1}{2}x\rfloor$ denotes the greatest integer less than or equal to $x/2$. $x\#y$ is defined to be $2^{|x|\cdot|y|}$.

We will frequently abbreviate $x\cdot y$ as xy. Also $A\leftrightarrow B$ is an abbreviation for the formula $(A\supset B)\wedge(B\supset A)$. So $\leftrightarrow$ is not a symbol in our first order logic.

We are using a larger set of non-logical symbols than is usually used for Peano arithmetic. This is partly to make it easy to define axiomatizations of fragments of Bounded Arithmetic. However, the $\#$ function (pronounced "smash", see Nelson [19]) has a more important role. The growth rate of $\#$ is exactly what we need to define functions in the polynomial hierarchy. Since $1\#x=2^{|x|}$ and $|\lfloor\frac{1}{2}(x\#y)\rfloor|=|x|\cdot|y|$, we can use $\#$, $\lfloor\frac{1}{2}x\rfloor$, and $\cdot$ to write the term $2^{p(|x|)}$ where p is any polynomial with non-negative coefficients. As we saw in Chapter 1, this is important for defining the polynomial hierarchy. Conversely, the value of any term of Bounded Arithmetic is bounded above by $2^{p(|x|)}$ for some suitable polynomial p.

We could generalize $\#$ as follows (see Hook [16]). Define $\#_2 = \#$. For $i\geq 2$, define $\#_{i+1}$ to be the binary function satisfying

$$x \#_{i+1} y \;=\; 2^{|x| \#_i |y|}$$

We could now add $\#_i$ to the language of arithmetic. Clearly, doing so would give us functions which have a larger than polynomial growth rate. In fact we could replace $\#$ by $\#_i$ everywhere in this dissertation and obtain analogous results *except* that instead of using polynomial time

Turing machines, we would use Turing machines with runtime bounded by terms involving $\#_i$. However we will not do this and we do **not** include $\#_3$, $\#_4$, ... in the language of Bounded Arithmetic.

Using 0, S, $+$, and $\cdot$ we can construct terms to denote natural numbers. For example, both $SSS0$ and $(SS0)+(S0)$ are terms which denote the number 3. There are two canonical formats for terms which denote numbers. First, $S^{(k)}0$ is the term with k applications of the successor function to 0; this term has value k. Second, I_k is a term with value k defined inductively by

$$\begin{aligned} I_0 &= 0 \\ I_{2k+1} &= I_{2k}+(S0) \\ I_{2(k+1)} &= (SS0)\cdot(I_{k+1}) \end{aligned}$$

Note that the length of the term I_k is proportional to the length $|k|$ of the binary representation of k; this is not true of $S^{(k)}0$. This will be important later when we arithmetize the syntax of Bounded Arithmetic in Chapter 8.

We shall frequently use integers in formulae. The integer is intended to be replaced by any closed term with value equal to the integer. Usually it makes no difference which term is used.

Definition: Quantifiers of the form $(\forall x)$ and $(\exists x)$ are called *unbounded quantifiers*. A *bounded quantifier* is one of the form $(\forall x{\le}t)$ or $(\exists x{\le}t)$ where t is any term not involving x. A *sharply bounded quantifier* is a bounded quantifier of the form $(\forall x{\le}|t|)$ or $(\exists x{\le}|t|)$ where again t is any term not involving x.

For the time being we will implicitly enlarge the syntax of first order logic to incorporate bounded quantifiers. In Chapter 4 we shall give an explicit and precise description of how bounded quantifiers are treated in first order logic. We shall do this by defining a natural deduction system with inferences for bounded quantifiers. The main result of Chapter 4 will be a cut elimination theorem which allows us to eliminate unbounded quantifiers from proofs of bounded formulae. Thus our main interest will be in first order logic without unbounded quantifiers.

A *bounded formula* is a formula with no unbounded quantifiers. We define a hierarchy of bounded formulae as follows:

Definition: The following sets of formulae are defined by induction on the complexity of formulae:

(1) $\Pi_0^b = \Sigma_0^b = \Delta_0^b$ is the set of formulae all of whose quantifiers are sharply bounded.

(2) Σ_{k+1}^b is defined inductively by:
  (a) $\Sigma_{k+1}^b \supseteq \Pi_k^b$
  (b) If A is in Σ_{k+1}^b then so are $(\exists x{\le}t)A$ and $(\forall x{\le}|t|)A$.
  (c) If $A,B\in\Sigma_{k+1}^b$ then $A\wedge B$ and $A\vee B$ are in Σ_{k+1}^b.

(d) If $A \in \Sigma_{k+1}^b$ and $B \in \Pi_{k+1}^b$ then $\neg B$ and $B \supset A$ are in Σ_{k+1}^b.

(3) Π_{k+1}^b is defined inductively by:

(a) $\Pi_{k+1}^b \supseteq \Sigma_k^b$

(b) If A is in Π_{k+1}^b then so are $(\forall x \le t)A$ and $(\exists x \le |t|)A$.

(c) If $A, B \in \Pi_{k+1}^b$ then $A \wedge B$ and $A \vee B$ are in Π_{k+1}^b.

(d) If $A \in \Pi_{k+1}^b$ and $B \in \Sigma_{k+1}^b$ then $\neg B$ and $B \supset A$ are in Π_{k+1}^b.

(4) Σ_{k+1}^b and Π_{k+1}^b are the smallest sets which satisfy (1)-(3).

Thus Σ_k^b and Π_k^b are defined analogously to the arithmetic hierarchy Σ_k^0 and Π_k^0 with bounded and sharply bounded quantifiers playing the roles of unbounded and bounded quantifiers respectively. That is, we count the alternations of bounded quantifiers ignoring the sharply bounded quantifiers. Bounded quantifiers have the following quantifier exchange property: let A be any formula, then

$$(\forall x \le |s|)(\exists y \le t)A(x,y) \Longleftrightarrow (\exists w \le (2s+1)\#(4(2t+1)^2))(\forall x \le |s|)(A(x,\beta(x+1,w)) \wedge \beta(x+1,w) \le t).$$

Essentially, w is a sequence which codes the values of y for each value of x. We have not yet defined the β function in Bounded Arithmetic and obviously the quantifier bound for w depends on the precise definition of β; however, the use of the $\#$ function is unavoidable. The $\#$ function has precisely the growth rate necessary to make this quantifier exchange property hold; this is part of the reason we feel that using the $\#$ function in Bounded Arithmetic is natural and elegant.

2.2. Axiomatizations of Bounded Arithmetic.

Peano arithmetic is normally axiomatized by a small number of open axioms and an induction schema. We shall form the axioms for Bounded Arithmetic by increasing the number of open axioms and severely restricting the induction axioms.

Definition: *BASIC* is a finite set of true open formulae of arithmetic which are sufficient to define the simple properties relating the function and predicate symbols of Bounded Arithmetic. *BASIC* consists of the following 32 formulae:

(1) $y \le x \supset y \le Sx$

(2) $x \ne Sx$

(3) $0 \le x$

(4) $x \le y \wedge x \ne y \leftrightarrow Sx \le y$

(5) $x \ne 0 \supset 2 \cdot x \ne 0$

(6) $y \le x \vee x \le y$

(7) $x \le y \wedge y \le x \supset x = y$

(8) $x \le y \wedge y \le z \supset x \le z$

(9) $|0| = 0$

(10) $x\neq 0\supset|2\cdot x|=S(|x|)\wedge|S(2\cdot x)|=S(|x|)$
(11) $|S0|=S0$
(12) $x\leq y\supset|x|\leq|y|$
(13) $|x\#y|=S(|x|\cdot|y|)$
(14) $0\#y=S0$
(15) $x\neq 0\ \supset\ 1\#(2\cdot x)=2(1\#x)\ \wedge\ 1\#(S(2\cdot x))=2(1\#x)$
(16) $x\#y=y\#x$
(17) $|x|=|y|\supset x\#z=y\#z$
(18) $|x|=|u|+|v|\supset x\#y=(u\#y)\cdot(v\#y)$
(19) $x\leq x+y$
(20) $x\leq y\wedge x\neq y\ \supset\ S(2\cdot x)\leq 2\cdot y\wedge S(2\cdot x)\neq 2\cdot y$
(21) $x+y=y+x$
(22) $x+0=x$
(23) $x+Sy=S(x+y)$
(24) $(x+y)+z=x+(y+z)$
(25) $x+y\leq x+z\leftrightarrow y\leq z$
(26) $x\cdot 0=0$
(27) $x\cdot(Sy)=(x\cdot y)+x$
(28) $x\cdot y=y\cdot x$
(29) $x\cdot(y+z)=(x\cdot y)+(x\cdot z)$
(30) $x\geq S0\supset(x\cdot y\leq x\cdot z\leftrightarrow y\leq z)$
(31) $x\neq 0\supset|x|=S(|\lfloor\frac{1}{2}x\rfloor|)$
(32) $x=\lfloor\frac{1}{2}y\rfloor\leftrightarrow(2\cdot x=y\vee S(2\cdot x)=y)$

(We are using 1 and 2 as abbreviations for the terms $S0$ and $SS0$.) Except for the results of Chapter 6, the precise definition of *BASIC* is not too important; any sufficiently large set of true open formulae would suffice. However, for the sake of definiteness, *BASIC* is defined to be the above 32 axioms. It will be important in Chapters 7 and 8 that *BASIC* is a finite set (or at least a polynomial time recognizable set).

In addition to the axioms in *BASIC*, we have various types of induction axioms.

Definition: Let Ψ be a set of formulae. The Ψ-*IND* axioms are:

$$A(0)\wedge(\forall x)(A(x)\supset A(Sx))\supset(\forall x)A(x)$$

where A is any formula in Ψ.

The Ψ-*PIND* axioms are:

$$A(0)\wedge(\forall x)(A(\lfloor\tfrac{1}{2}x\rfloor)\supset A(x))\supset(\forall x)A(x)$$

where A is any formula in Ψ.

The Ψ*–LIND* axioms are:

$$A(0)\wedge(\forall x)(A(x)\supset A(S\dot{x}))\supset(\forall x)A(|x|)$$

where A is any formula in Ψ.

A little reflection yields the intuitive feeling that Ψ*–IND* is stronger than Ψ*–PIND*. For example, suppose we know $A(0)$ is true and we wish to deduce that $A(100)$ is true. If we use Ψ*–IND* we will deduce $A(1)$ from $A(0)$, then $A(2)$ from $A(1)$, and so on for 100 steps. On the other hand, Ψ*–PIND* deduces $A(1)$, then $A(3)$, $A(6)$, $A(12)$, $A(25)$, $A(50)$, and finally $A(100)$. Thus the Ψ*–IND* axiom "automated" 100 inferences, whereas the Ψ*–PIND* automated only 7 inferences. Since the conclusions of Ψ*–IND* and Ψ*–PIND* are the same we conclude that the hypothesis of Ψ*–PIND* is stronger than the hypothesis of Ψ*–IND* and hence we feel that the Ψ*–PIND* axioms are weaker than the Ψ*–IND* axioms. We shall prove this properly below.

This is a good place to mention explicitly that we do **not** have the function $x\mapsto 2^x$ in Bounded Arithmetic. Hence the conclusion $(\forall x)A(|x|)$ of Ψ*–LIND* is weaker than $(\forall x)A(x)$. Indeed, in a nonstandard model for Bounded Arithmetic the function $x\mapsto 2^x$ may not be total and hence $x\mapsto|x|$ may not be onto.

Definition: The following theories are fragments of Bounded Arithmetic. Each theory has the language of arithmetic defined in §2.1.

(1) S_2^i has axioms:
  (a) *BASIC* axioms
  (b) Σ_i^b*–PIND* axioms.

(2) T_2^i has axioms:
  (a) *BASIC* axioms
  (b) Σ_i^b*–IND* axioms.

(3) S_2 is $\bigcup_i S_2^i$.

(4) T_2 is $\bigcup_i T_2^i$.

(5) $S_2^{(-1)}$ is the theory with only the *BASIC* axioms. $T_2^{(-1)}$ is the same theory.

Later we shall show that $T_2^i\vdash S_2^i$ and $S_2^i\vdash T_2^{i-1}$ where $i>0$. The theories we are most interested in are S_2^i, as these fragments of Bounded Arithmetic have the nicest properties. Most of this dissertation is concerned with the theories S_2^i. The subscript "2" denotes the presence of the $\#$ function. In general, for $k\geq 1$, S_k^i is defined like S_2^i but with the function symbols $\#_j$ for all $2\leq j\leq k$ and with their defining axioms.

Proposition 1: Ψ–*IND* $\implies$ Ψ–*LIND*.

Proof: The hypotheses of Ψ–*IND* and Ψ–*LIND* are the same and the conclusion of Ψ–*IND* is stronger than the conclusion of Ψ–*LIND*. □

2.3. Introducing Function and Predicate Symbols.

Bounded Arithmetic is powerful enough to define many functions besides the six functions in the formal language. It is generally true that whenever a theory can define a function, a conservative extension is obtained by augmenting the language to include a new function symbol for the defined function. We shall be especially interested in introducing function symbols which can be used in formulae in induction axioms.

Definition: Let R be a fragment of Bounded Arithmetic. Suppose A is a Σ_i^b–formula and that

$$R \vdash (\forall \vec{x})(\exists y \leq t) A(\vec{x},y)$$

and

$$R \vdash (\forall \vec{x})(\forall y)(\forall z)(A(\vec{x},y) \wedge A(\vec{x},z) \supset y=z).$$

Then we say that R can Σ_i^b–*define* the function f such that $(\forall \vec{x})A(\vec{x},f(\vec{x}))$ is satisfied. (It should be noted that the above definition makes sense only if $\vec{x}$ and y are all the free variables of A; if not, enlarge $\vec{x}$ to include the rest of them.)

Definition: Let f be a new function symbol. We define $\Delta_0^b(f)$, $\Sigma_i^b(f)$ and $\Pi_i^b(f)$ to be sets of bounded formulae in the language of Bounded Arithmetic plus the symbol f. These sets of formulae are defined by counting alternations of bounded quantifiers, ignoring the sharply bounded quantifiers, exactly as in the definition of Δ_0^b, Σ_i^b and Π_i^b.

If p is a new predicate symbol we define $\Delta_0^b(p)$, $\Sigma_i^b(p)$ and $\Pi_i^b(p)$ similarly.

Theorem 2: Let R be a fragment of Bounded Arithmetic. Suppose R can Σ_1^b–define the function f. Let R^* be the theory obtained from R by adding f as a new function symbol and adding the defining axiom for f. Then, if $i>0$ and B is a $\Sigma_i^b(f)$– (or a $\Pi_i^b(f)$–) formula, there is a $B^* \in \Sigma_i^b$ (or Π_i^b, respectively) such that $R^* \vdash B^* \leftrightarrow B$.

Proof: The defining axiom for f is

$$f(\vec{x})=y \leftrightarrow A(\vec{x},y)$$

where A is a Σ_1^b–formula. Let B be a bounded formula containing the symbol f. We first define the formula B_1 as follows: suppose f occurs in a term which bounds a quantifier, say $(Qx \leq s)D$ is a subformula of B where the term s involves f. Replace each occurrence of $f(\vec{r})$ in

s by the term $t(\vec{r})$. (t is the bound in the Σ_1^b–definition of f, see the definition above.) This yields a term s'. Now, $(\exists x \leq s)D$ is provably equivalent to $(\exists x \leq s')(x \leq s \wedge D)$ and $(\forall x \leq s)D$ is provably equivalent to $(\forall x \leq s')(x \leq s \supset D)$. By repeating this procedure, we can form B_1 so that

(1) $R^* \vdash B \leftrightarrow B_1$, and

(2) B_1 does not contain f appearing in any term which bounds a quantifier.

We next obtain a formula B_2 in prenex normal form by applying prenex operations to B_1 so that $R^* \vdash B_2 \leftrightarrow B_1$. Furthermore, if B is a Σ_i^b– (or a Π_i^b–) formula, then so are B_1 and B_2.

Let the mantissa of B_2 be D; that is to say, suppose

$$B_2 \;=\; (Q_1 x_1 \leq s_1) \cdots (Q_n x_n \leq s_n) D$$

where D is an open formula. Let $f(\vec{r})$ be a term appearing in D. Obtain D' by replacing $f(\vec{r})$ everywhere in D by a new variable z. Define

$$D_A \;=\; (\forall z \leq t(\vec{r}))(A(\vec{r},z) \supset D')$$

and

$$D_E \;=\; (\exists z \leq t(\vec{r}))(A(\vec{r},z) \wedge D').$$

Let $D^\forall$ and $D^\exists$ be their respective prenex normal forms. Then $D^\forall$ is a $\Pi_1^b(f)$–formula and $D^\exists$ is a $\Sigma_1^b(f)$–formula, and

$$R^* \vdash (D \leftrightarrow D^\forall) \wedge (D \leftrightarrow D^\exists).$$

Define B_3 from B_2 by replacing the mantissa D by either $D^\forall$ or $D^\exists$, whichever is appropriate. We can do this so that B_3 has the same alternation of (non-sharply) bounded quantifiers as B_2. Also,

$$R^* \vdash B_3 \leftrightarrow B_2.$$

B_3 was formed from B_2 so that all occurrences of the term $f(\vec{r})$ were eliminated. By iterating this procedure, we obtain B_4 from B_3, B_5 from B_4, and so on, until all occurrences of f have been eliminated. We let B^* be the B_i such that $i \geq 2$ and f does not appear in B_i.

Q.E.D. □

Corollary 3: Let R be one of the theories S_2^i or T_2^i (where $i \geq 1$). Suppose $f_1, \ldots, f_k$ are functions Σ_1^b-definable by R. Let $\hat{R}$ be the theory obtained from R by including new function symbols $f_1, \ldots, f_k$ and their defining axioms and including all $\Sigma_i^b(\vec{f})$-*PIND* axioms or $\Sigma_i^b(\vec{f})$-*IND* axioms (respectively). Then $\hat{R}$ is a conservative extension of R.

Proof: Form R^* by adjoining $f_1, \ldots, f_k$ and adding their defining axioms. It is well known that R^* is a conservative extension of R. Now, by Theorem 2, each $\Sigma_i^b(\vec{f})$-formula is provably (in R^*) equivalent to a Σ_i^b-formula. Thus $R^* \vdash \Sigma_i^b(\vec{f})$-*PIND* (or $\Sigma_i^b(\vec{f})$-*IND* respectively). Hence $\hat{R} \equiv R^*$. □

The upshot of the last theorem is that we may freely adjoin Σ_1^b-definable functions to any fragment of Bounded Arithmetic and use these function symbols without restriction in induction formulae.

We can also define a similar condition for introducing new relation symbols:

Theorem 4: Let R be a fragment of Bounded Arithmetic. Suppose A and B are Σ_1^b and Π_1^b formulae, respectively. Also suppose $R \vdash A \leftrightarrow B$. Let R^* be the theory obtained from R by adjoining a new predicate symbol p and the defining axiom

$$p(\vec{x}) \leftrightarrow A(\vec{x}).$$

($\vec{x}$ must include all the free variables of A.)

Then R^* is a conservative extension of R and if $i \geq 1$ and C is any $\Sigma_i^b(p)$- or $\Pi_i^b(p)$-formula then there is a Σ_i^b- or Π_i^b-formula C^* (respectively) such that $R^* \vdash C \leftrightarrow C^*$.

Proof: Similar to the proof of Theorem 2. □

It is convenient to have a name for predicates which satisfy the conditions of Theorem 4:

Definition: Let R be a theory and A be any formula. We say that A is Δ_i^b with respect to the theory R iff there are formulae $B \in \Sigma_i^b$ and $C \in \Pi_i^b$ such that $R \vdash A \leftrightarrow B$ and $R \vdash A \leftrightarrow C$.

When it is clear which theory R is being discussed, we shall merely say A is Δ_i^b when we mean A is Δ_i^b with respect to R.

It follows immediately from Theorem 4 that if A is a Δ_1^b-formula, then a new predicate symbol p can be introduced with the defining axiom $p(\vec{x}) \leftrightarrow A(\vec{x})$ and that p can be used freely in formulae in induction axioms. Thus we have established conditions for introducing new function and predicate symbols into a fragment of Bounded Arithmetic, so that the new symbols can be used in induction axioms.

Example: We define the binary subtraction function $\dot{-}$ as

$$x\dot{-}y=z \iff y+z=x\lor(z=0\land x\leq y)$$

We show that $\dot{-}$ can be Σ_1^b–defined in T_2^1. To do this we have to show that T_2^1 can prove

$$(\forall x)(\forall y)(\exists z\leq x)M(x,y,z)$$

and

$$M(x,y,z)\land M(x,y,z')\supset z=z'$$

where $M(x,y,z)$ is the formula on the righthand side of the defining axiom for $\dot{-}$.

The second formula to be proved is the uniqueness condition. This follows directly from the *BASIC* axioms without the use of any induction axioms.

To prove the existence condition, we will need to use the induction axioms. It is not hard to prove the following formulae in T_2^1:

$$(\exists z\leq 0)M(0,y,z)$$

$$(\exists z\leq x)M(x,y,z)\supset(\exists z\leq Sx)M(Sx,y,z).$$

From these two formulae we use Σ_1^b–*IND* to derive

$$(\forall x)(\forall y)(\exists z\leq x)M(x,y,z).$$

Thus the subtraction function can be defined in T_2^1.

We will later show by a much more complicated argument that the $\dot{-}$ function can be Σ_1^b–defined in S_2^1 as well.

As an application of the above example, we show that the theory T_2^i can derive the Π_i^b–*IND* axioms.

Theorem 5: The Π_i^b–*IND* axioms are theorems of T_2^i if $i\geq 1$.

Proof: Let A be a Π_i^b–formula. We want to show

$$T_2^i\vdash A(0)\land(\forall x)(A(x)\supset A(Sx))\supset(\forall x)A(x).$$

Let $B(x,y)$ be the Σ_i^b–formula $\neg A(y\dot{-}x)$. Then

$$T_2^i \vdash B(0,y) \wedge (\forall x)(B(x,y) \supset B(Sx,y)) \supset B(y,y)$$

or, equivalently,

$$T_2^i \vdash \neg A(y) \wedge (\forall x)(\neg A(y \dot{-} x) \supset \neg A(y \dot{-} Sx)) \supset \neg A(0).$$

From this we can readily derive

$$T_2^i \vdash A(0) \wedge (\forall x)(A(x) \supset A(Sx)) \supset A(y).$$

Taking the universal closure of this last formula proves the desired induction axiom.

Q.E.D. □

2.4. Bootstrapping S_2^1 - Phase 1.

The term "bootstrapping" is a computer term which describes the process of starting the operations of a computer. It used to be common to power up a computer with only a small amount of software loaded, say about 80 bytes, the amount of data which fits on a Hollerith card. This small amount of software would be responsible for reading from tape or cards the entire operating system, thus making the computer fully operational. This process was called "bootstrapping" from the analogy of "lifting oneself by the bootstraps."

Similarly we need to bootstrap S_2^1. That is, we shall have to do a lot of work to define some simple functions and predicates in S_2^1 (for example, subtraction). Once we have completed the bootstrapping it will be easy to show that S_2^1 is actually a fairly strong system which can define a variety of functions and predicates.

To a certain extent, our bootstrapping of S_2^1 is recapitulating the work of Ed Nelson [19], Wilkie-Paris [31] and Wilmers [32]. However, [19] and [31] work in the theory S_2 not S_2^1, and they are consequently only concerned about defining functions and predicates with arbitrary bounded formulae. For us, it is very important that functions be Σ_1^b–defined and predicates be Δ_1^b–defined. Wilmers [32] does use a very weak fragment of S_2 but his work does not seem to apply to S_2^1.

Before we begin the bootstrapping of S_2^1 we show that the Σ_1^b–*LIND* axioms can be derived in S_2^1.

Theorem 6: Let $i \geq 0$. The Σ_i^b–*LIND* axioms are theorems of S_2^i.

Proof: Let A be a Σ_i^b–formula. We want to show that

$$S_2^i \vdash A(0) \wedge (\forall x)(A(x) \supset A(Sx)) \supset (\forall x)A(|x|).$$

Let $B(x)$ be the formula $A(|x|)$. Then

$$S_2^i \vdash A(0) \supset B(0)$$

and

$$S_2^i \vdash (\forall x)(A(x) \supset A(Sx)) \supset (\forall x)(B(\lfloor \tfrac{1}{2}x \rfloor) \supset B(x)).$$

But B is a Σ_i^b-formula so by the use of the Σ_i^b-*PIND* axiom for B we get

$$S_2^i \vdash A(0) \wedge (\forall x)(A(x) \supset A(Sx)) \supset (\forall x) B(x)$$

which is what we wanted to show.

Q.E.D. □

We bootstrap S_2^1 by showing that the following functions and predicates are Σ_1^b-definable in S_2^1 and are Δ_1^b with respect to S_2^1, respectively.

(a) We introduce one predicate and two functions by:

$$a < b \iff a \le b \wedge a \ne b$$
$$c = \max(a,b) \iff (c \ge a \wedge c = b) \vee (c \ge b \wedge c = a)$$
$$c = \min(a,b) \iff (c \le a \wedge c = b) \vee (c \le b \wedge c = a)$$

The uniqueness and existence conditions for these functions follow easily from the *BASIC* axioms without any use of induction. Since the defining formula for $a < b$ is open it is trivially Δ_1^b.

(b) The predecessor function is an inverse to the successor function defined by

$$b = P(a) \iff (a = 0 \wedge b = 0) \vee Sb = a.$$

The uniqueness condition for this definition follows easily from the *BASIC* axioms without any use of induction axioms. For existence, let $M(a,b)$ be the defining equation for $P(a) = b$; then we can prove

$$(\exists z \le 0) M(0,z)$$

and

$$(\exists z \le \lfloor \tfrac{1}{2}x \rfloor) M(\lfloor \tfrac{1}{2}x \rfloor, z) \supset (\exists z \le x) M(x,z)$$

from the *BASIC* axioms again without any use of induction axioms. Finally, Σ_1^b-*PIND* yields

$$S^1_2 \vdash (\forall x)(\exists z \le x) M(x,z).$$

(c) $Power2(a) \iff S(|P(a)|)=|a|$

This predicate symbol is clearly Δ^b_1 with respect to S^1_2. Moreover, S^1_2 can prove many nice properties of *Power2*. In particular, S^1_2 can prove the following formulae:

$$Power2(a) \supset a \neq 0$$
$$Power2(a) \supset Power2(a+a)$$
$$Power2(a) \wedge |a| \le |c| \supset a \le c$$
$$Power2(a) \wedge Power2(b) \wedge |a|=|b| \supset a=b$$
$$Power2(a) \wedge a \neq 1 \supset Power2(\lfloor \tfrac{1}{2} a \rfloor)$$

For example, the fourth formula follows from the open formula

$$a=Sc \wedge b=Sd \wedge |a|=S(|c|) \wedge |b|=S(|d|) \wedge |a|=|b| \supset a=b$$

which in turn can be proved in S^1_2 (without the use of any induction axioms.) We leave to the reader the verification of our claim that the other four formulae are also theorems of S^1_2.

(d) We can define an exponentiation function with restricted range by:

$$c=Exp(a,b) \iff Power2(c) \wedge |c|=1+\min(|b|,a)$$

or informally, $Exp(a,b) = 2^{\min(|b|,a)}$.

Let $M(a,b,c)$ be the formula on the righthand side of the definition of *Exp*. Then, by the properties of *Power2* discussed above,

$$S^1_2 \vdash M(x,y,z) \wedge M(x,y,u) \supset z=u.$$

Also,

$$S^1_2 \vdash M(x,y,z) \wedge x < |y| \supset M(Sx,y,2z)$$
$$S^1_2 \vdash M(x,y,z) \wedge x \ge |y| \supset M(Sx,y,z)$$
$$S^1_2 \vdash M(x,y,z) \supset z \le 2y+1$$

From this we get

$$S_2^1 \vdash (\exists z \le 2y+1)M(x,y,z) \supset (\exists z \le 2y+1)M(Sx,y,z).$$

Hence,

$$S_2^1 \vdash (\forall x \le |y|)(\exists z \le 2y+1)(x \le u \supset M(x,y,z)) \supset (\forall x \le |y|)(\exists z \le 2y+1)(x \le Su \supset M(x,y,z)).$$

On the other hand, $S_2^1 \vdash (\forall y)M(0,y,1)$ so by Σ_1^b-$LIND$ with respect to the variable u (remember, we don't count sharply bounded quantifiers):

$$S_2^1 \vdash (\forall x \le |y|)(\exists z \le 2y+1)(x \le |y| \supset M(x,y,z))$$

and hence

$$S_2^1 \vdash (\forall x \le |y|)(\exists z \le 2y+1)M(x,y,z).$$

And since $S_2^1 \vdash x \ge |y| \wedge M(|y|,y,z) \supset M(x,y,z)$, we have

$$S_2^1 \vdash (\forall x)(\exists z \le 2y+1)M(x,y,z).$$

This is what we needed to demonstrate that Exp is properly defined in S_2^1.

It is important to note that we have **not** defined exponentiation, but only a restricted exponentiation. Indeed, in the formula

$$z = Exp(i,y) = 2^{\min(i,|y|)}$$

the argument y is a "dummy variable" whose sole purpose is to restrict the range of the function. Frequently we shall simplify our notation and write 2^i as a function when it is provably well defined; for instance, we would write $(\forall i \le |x|)B(2^i)$ instead of the more correct $(\forall i \le |x|)B(2^{\min(i,|x|)})$.

(e) $b = Mod2(a) \iff b + 2 \cdot \lfloor \frac{1}{2} a \rfloor = a$

$Mod2(a)$ is either zero or one depending on whether a is even or odd, respectively. We can easily prove the necessary uniqueness and existence conditions from the *BASIC* axioms.

(f) We define functions for obtaining the "less significant part" and the "more significant part" by defining the following predicate and functions:

$$Decomp(a,b,c,d) \iff |c| \le b \wedge d \cdot 2^{\min(|a|,b)} + c = a$$

$$c = LSP(a,b) \iff (\exists d \le a)Decomp(a,b,c,d)$$

$$d = MSP(a,b) \iff (\exists c \le a)Decomp(a,b,c,d)$$

Clearly, *Decomp* is Δ_1^b–defined. Also, is not difficult to see that

$$S_2^1 \vdash Decomp(a,b,c,d) \wedge Decomp(a,b,e,f) \supset c=e \wedge d=f.$$

This establishes the uniqueness conditions for both *MSP* and *LSP*.

It remains to show that the existence conditions hold; namely, that

$$S_2^1 \vdash (\exists c \leq a)(\exists d \leq a) Decomp(a,b,c,d).$$

Since $S_2^1 \vdash Decomp(a,0,0,a)$, we know that

$$S_2^1 \vdash (\forall x \leq 0)(\exists c \leq a)(\exists d \leq a) Decomp(a,x,c,d).$$

Also, the following are provable in S_2^1:

$$x < |a| \wedge Decomp(a,x,c,d) \supset Decomp(a,Sx,c+2^x \cdot Mod2(d), \lfloor \tfrac{1}{2} d \rfloor)$$

and

$$Decomp(a,b,c,d) \supset c \leq a \wedge d \leq a.$$

It follows readily that

$$S_2^1 \vdash (\forall x \leq |a|)(\exists c \leq a)(\exists d \leq a)(x \leq u \supset Decomp(a,x,c,d)) \supset$$
$$\supset (\forall x \leq |a|)(\exists c \leq a)(\exists d \leq a)(x \leq Su \supset Decomp(a,x,c,d)).$$

From this, by use of Σ_1^b–*LIND*

$$S_2^1 \vdash (\forall x \leq |a|)(\exists c \leq a)(\exists d \leq a) Decomp(a,x,c,d).$$

Since $S_2^1 \vdash x \geq |a| \supset Decomp(a,x,a,0)$, this suffices to prove the existence condition.

(g) $c = Bit(b,a) \iff c = Mod2(MSP(a,b))$

So $Bit(b,a)$ is the value of the bit in the 2^b position of the binary representation of a. Since *Bit* is defined as the composition of functions already introduced in S_2^1 it is clear that *Bit* is Σ_1^b–defined.

An important property which is provable in S_2^1 is:

$$|a| \geq |b| \wedge (\forall y < |a|)(Bit(y,a) = Bit(y,b)) \supset a = b.$$

That is, it is provable that the binary representation of a number uniquely determines the number. This can be proved in S_2^1 by using Σ_1^b–*LIND* with respect to the variable u on the formula

$$(\forall y<u)(Bit(y,a)=Bit(y,b))\supset LSP(a,u)=LSP(b,u).$$

The details are left to the reader.

Further note that S_2^1 can prove all the simple relationships between *Bit*, *MSP* and *LSP*; for example,

$$S_2^1\vdash Bit(b,a)=MSP(LSP(a,Sb),b)$$

$$S_2^1\vdash Bit(b,a)=LSP(MSP(a,b),1)$$

$$S_2^1\vdash Bit(b+c,a)=Bit(c,MSP(a,b)).$$

(h) Before we can define the subtraction function, we need a restricted version of subtraction:

$$c=LENMINUS(a,b) \iff (b\le|a|\wedge c+b=|a|)\vee(b\ge|a|\wedge c=0)$$

So $LENMINUS(a,b)$ is equal to $|a|\dot{-}b$, or in other words, *LENMINUS* is a subtraction function with domain restricted to very small numbers. The uniqueness condition is easy to prove from the *BASIC* axioms. Because the function is restricted we are able to prove the existence condition with induction on Δ_0^b–formulae. It will suffice to show that

$$S_2^1\vdash(\forall x\le|a|)(\exists y\le|a|)(x+y=|a|).$$

Now,

$$S_2^1\vdash x<|a|\wedge x+y=|a|\supset S(x)+P(y)=|a|.$$

So,

$$S_2^1\vdash(\forall x\le|a|)(\exists y\le|a|)(x\le u\supset x+y=|a|)\supset(\forall x\le|a|)(\exists y\le|a|)(x\le Su\supset x+y=|a|).$$

By Σ_1^b–*LIND* we obtain the desired result.

(i) Finally, we show that subtraction can be Σ_1^b–defined in S_2^1.

$$c=a\dot{-}b \iff a+b=c\vee(c=0\wedge a<b)$$

The uniqueness condition for subtraction is immediate from the *BASIC* axioms. The existence condition is not too hard now that we have defined *MSP* and *LENMINUS*; we will use Σ_1^b–*LIND* on the formula $M(a,b,u)$ defined as

$$b \le a \supset (\exists x \le a)(x + MSP(b, |a| \dot{-} u) = MSP(a, |a| \dot{-} u)).$$

Here we are using $|a| \dot{-} u$ as an abbreviation for $LENMINUS(a,u)$. Now,

$$S_2^1 \vdash |b| \le |a| \supset MSP(b, |a|) = 0.$$

So $S_2^1 \vdash M(a,b,0)$. Also, $S_2^1 \vdash M(a,b,u) \supset M(a,b,Su)$ can be proved without too much difficulty; this follows from the fact that S_2^1 can prove all of the following:

(i) $b \le a \supset MSP(b,z) < MSP(a,z) \vee (MSP(b,z) = MSP(a,z) \wedge LSP(b,z) \le LSP(a,z))$

(ii) $b \le a \wedge x + MSP(b,Sz) = MSP(a,Sz) \wedge Bit(z,a) = Bit(z,b) \supset$
$\supset 2x + MSP(b,z) = MSP(a,z)$

(iii) $b \le a \wedge x + MSP(b,Sz) = MSP(a,Sz) \wedge Bit(z,a) > Bit(z,b) \supset$
$\supset (2x+1) + MSP(b,z) = MSP(a,z)$

(iv) $b \le a \wedge x + MSP(b,Sz) = MSP(a,Sz) \wedge Bit(z,a) < Bit(z,b) \supset$
$\supset (2x \dot{-} 1) + MSP(b,z) = MSP(a,z)$

By Σ_1^b-$LIND$, $S_2^1 \vdash M(a,b,|a|)$, which is equivalent to the existence condition for the definition of subtraction since $S_2^1 \vdash MSP(x,0) = x$.

(j) $QuoRem(a,b,c,d) \iff (b=0 \wedge c=0 \wedge d=0) \vee (d < b \wedge a = c \cdot b + d)$

$c = \lfloor a/b \rfloor \iff (\exists d \le b) QuoRem(a,b,c,d)$

$d = Rem(a,b) \iff (\exists c \le a) QuoRem(a,b,c,d)$

The uniqueness conditions are easily proved. The existence conditions can be proved by induction on the length of a; we leave this as an exercise for the reader. (Hint: how do you compute the quotient and remainder for $\frac{a}{b}$ if you know them for $\lfloor \frac{1}{2} a \rfloor / b$?)

(k) $b \mid a \iff Rem(a,b) = 0 \wedge b \neq 0$

(l) $Even(a) \iff Mod2(a) = 0$

$Odd(a) \iff Mod2(a) = 1$

(m) $Comma(b,a) \iff Even(b) \wedge Bit(b,a){=}1 \wedge Bit(Sb,a){=}0$

$c{=}Digit(b,a) \iff [Even(b) \wedge Bit(Sb,a){=}1 \wedge Bit(b,a){=}c] \vee [(Odd(b) \vee Bit(Sb,a){=}0) \wedge c{=}2]$

Comma and *Digit* are immediately seen to be Δ_1^b–definable and Σ_1^b–definable by S_2^1. They will be useful for defining an encoding for sequences. It is important to note that we will **not** be using the same encodings for sequences as we used in Chapter 1.

(n) $PSqSL(a,b,c) \iff |a|+2=2\cdot c\cdot Sb \wedge (\forall y<|a|)((2b+2) \mid (y+2) \supset Comma(y,a)) \wedge$
$\wedge (\forall y<|a|)(Even(y) \wedge \neg(2b+2) \mid (y+2) \supset Digit(y,a) \neq 2)$

$b{=}ProtoSize(a) \iff 2\cdot b \leq |a| \wedge (2\cdot b{=}|a| \vee Comma(2\cdot b,a)) \wedge (\forall x<b)(\neg Comma(2\cdot x,a))$

$c{=}ProtoLen(a) \iff c{=}\lfloor(|a|+2)/(2\cdot ProtoSize(a)+2)\rfloor$

$ProtoSeq(a) \iff PSqSL(a,ProtoSize(a),ProtoLen(a))$

These functions and relations define *protosequences*, and give us a primitive method of encoding sequences. Protosequences have the restriction that each element of the protosequence is coded by a fixed length code; if necessary, leading zeros are added to the element to pad it out to the required length. $PSqSL(a,b,c)$ asserts that a encodes a protosequence of c numbers, each of which is coded as a b-bit number and is preceded by a comma. The fact that a is such a protosequence can be verified by checking the positions of the "commas" in a. Note that there is no protosequence for the empty sequence.

We leave it to the reader to prove that *ProtoLen* and *ProtoSize* can be Σ_1^b–defined in S_2^1 and that *PSqSL* and *ProtoSeq* are Δ_1^b with respect to S_2^1.

(o) $c{=}Proto\beta(b,a) \iff (\neg ProtoSeq(a) \wedge c{=}0) \vee [ProtoSeq(a) \wedge |c| \leq ProtoSize(a) \wedge$
$\wedge (\forall y<ProtoSize(a))(Bit(y,c){=}Digit(2\cdot(y+(ProtoSize(a)+1)\cdot(b \dot{-} 1)),a))]$

So if $a{=}<a_1,\ldots,a_k>$ then $Proto\beta(i,a){=}a_i$. Note that (unlike the sequences used in Chapter 1) the numbers are not coded in bit-reversed order. The sequence is coded with a_1 coded by the low order bits of the binary representation of a and with a_k coded by the high order bits.

The uniqueness condition for $Proto\beta$ is a consequence of the fact that the binary representation of a number uniquely determines the number, which as we noted earlier is provable in S_2^1.

It is important to note that since $S_2^1 \vdash ProtoSize(a) \leq |a|$, the quantifier $(\forall y<ProtoSize(a))$ can be replaced by a sharply bounded quantifier. This makes it possible to prove the existence condition for $Proto\beta$ by using Σ_1^b–*LIND* with respect to the variable u on the formula

$$ProtoSeq(a) \wedge u < ProtoSize(a) \supset$$
$$\supset (\exists c \le a)\left[|c| \le u \wedge (\forall y \le u)(Bit(y,c) = Digit(2 \cdot (y + (ProtoSize(a)+1) \cdot (b \dot- 1)), a))\right].$$

S^1_2 also proves that protosequences exist. Indeed, it can be shown by induction on the length of a that

$$S^1_2 \vdash b \le |d| \wedge |a| \le b \supset (\exists z \le 4 \cdot d^2)(PSqSL(z,b,1) \wedge Proto\beta(1,z) = a).$$

(p) $c = ProtoStar(a,b) \iff (\neg ProtoSeq(a) \wedge c = 0) \vee [ProtoSeq(c) \wedge ProtoSeq(a) \wedge$
$\wedge ProtoSize(c) = ProtoSize(a) \wedge ProtoLen(c) = 1 + ProtoLen(a) \wedge$
$\wedge (\forall x < ProtoLen(a))(Proto\beta(z+1,a) = Proto\beta(z+1,c)) \wedge$
$\wedge Proto\beta(ProtoLen(a)+1,c) = LSP(b, ProtoSize(a))]$

$ProtoStar(a,b)$ is the Gödel number for the protosequence obtained by adding b as an additional element to the end of the protosequence coded by a. If b is too large to fit into the protosequence, only the less significant part of b is used.

We omit the proofs of the uniqueness and existence conditions for *ProtoStar*. The reader may supply them if desired.

2.5. Bootstrapping S^1_2 - Phase 2.

For the second phase of bootstrapping S^1_2 we wish to define sequences with variable length elements; these sequences will supercede the protosequences defined above. Some of the functions and predicates we wish to define are:

$Seq(w)$	true iff w is a valid sequence
$Size(w)$	the maximum of the lengths of entries of w
$Len(w)$	the number of elements in w
$\beta(i,w)$	the value of the i-th value of w
$*$	a function which adds a new element to the end of a sequence
$**$	a function which concatenates two sequences

It is not difficult to define *Seq*, *Size*, $*$ and $**$ since each of these can be defined by "local" operations. However, $Len(w)$ and β are harder to define. Computing $Len(w)$ involves counting the number of *Comma*'s in w and hence is a "global" operation. Likewise, to calculate $\beta(i,w)$, it is necessary to locate the i-th entry of w and this again requires counting.

Hence we are led to the following:

Definition: Let $A(z,y,\vec{x})$ be any formula. The function $f(y,\vec{x})$ is *defined by length bounded counting from A* iff f satisfies

$$f(y,\vec{x}) \;=\; (\#z\leq|y|)A(z,y,\vec{x})$$

where $(\#z\leq t)(\;\cdots\;)$ means "the number of $z\leq t$ such that $\;\cdots\;$".

Of course, we can define *bounded counting* in a similar way, except that the bound t need not be a length. Bounded counting has been investigated by Valiant [29] and it is an open problem whether functions defined by bounded counting are always in the polynomial hierarchy. Of course, any function which is definable by a bounded formula is in the polynomial hierarchy and thus we are not able to use bounded counting in S_2^1 (at least at our present state of knowledge). However, functions defined by bounded counting are computable by polynomial-space bounded Turing machines and in §10.2 we discuss how bounded counting may be defined in a second-order theory of Bounded Arithmetic.

Theorem 7: Let $A(z,y,\vec{x})$ be Δ_1^b with respect to S_2^1. Let f be the function defined by length bounded counting from A. Then f can be Σ_1^b–defined in S_2^1.

Proof: We introduce a new $(k+1)$-ary function symbol g defined by

$$c=g(b,\vec{a}) \iff |c|\leq|b|+1\wedge(\forall x\leq|b|)(Bit(x,c)=1\leftrightarrow A(x,b,\vec{a})).$$

The existence and uniqueness conditions from g are readily proved in S_2^1. For the existence condition we use Σ_1^b–*LIND* with respect to u on the formula

$$(\exists c\leq 2b+1)(\forall x\leq|b|+1)\left[x\leq u\supset(Bit(x,c)=1\leftrightarrow A(x,b,\vec{a}))\right].$$

Note that we needed the fact that A is Δ_1^b in order to Σ_1^b–define g.

We define the function *Numones*, which computes the number of ones in the binary representation of a number a, by

$$\begin{aligned} b=Numones(a) \iff\; & (\exists w\leq 2^{(2\cdot||a||+2)\cdot(|a|+1)})\left[PSqSL(w,||a||,|a|+1)\wedge\right.\\ & \wedge Proto\beta(1,w)=0\wedge b=Proto\beta(|a|+1,w)\wedge\\ & \left.\wedge(\forall i<|a|)(Proto\beta(i+1,w)+Bit(i,a)=Proto\beta(i+2,w))\right]. \end{aligned}$$

The uniqueness and existence conditions for *Numones* are provable in S_2^1 by induction on the length of a - we omit the details. Note that the use of the # function is required to express the bound on w in the defining equation of *Numones*; this is the first time we have used the #

function for bootstrapping S_2^1.

We can now define f as

$$f(b,\vec{a}) \;=\; Numones(g(b,\vec{a})).$$

Q.E.D. □

Theorem 8: S_2^1 proves the following:

$$Numones(x)+Numones(y)=Numones(x\cdot 2^{|y|}+y).$$

Proof: It is not hard to prove this using Σ_1^b–*LIND*. This depends on the fact that S_2^1 can prove simple properties about *Bit*, *MSP* and *LSP*; see §2.4(g). □

Theorem 9: The following functions are Σ_1^b–definable in S_2^1 and hence can be introduced as defined function symbols.

(i) $f_1(\vec{x}) = \min\{t(y){:}y\le|s|\}$

(ii) $f_2(\vec{x}) = \max\{t(y){:}y\le|s|\}$

(iii) $f_3(\vec{x}) = (\mu y\le|s|)A(y)$

where s and t are terms and A is a Δ_1^b–formula. The free variables of s are the $\vec{x}$; the free variables of t and of A may include y and $\vec{x}$. The terms s and t may involve Σ_1^b–defined function symbols.

Proof: The existence and uniqueness conditions for f_1 and f_2 can be proved easily by using Σ_1^b–*LIND* on the length of s. We can define f_3 in terms of *Numones* by

$$f_3(\vec{x}) = (\#y\le|s|)(\forall z\le|s|)(z\le y\supset\neg A(z)).$$

□

Lemma 10: Let f_1, f_2, and f_3 be the function symbols introduced in Theorem 9. Then S_2^1 proves the following:

(i) $(\exists y \leq |s|)(f_1(\vec{x})=t(y,\vec{x})) \wedge (\forall y \leq |s|)(f_1(\vec{x}) \leq t(y,\vec{x}))$

(ii) $(\exists y \leq |s|)(f_2(\vec{x})=t(y,\vec{x})) \wedge (\forall y \leq |s|)(f_2(\vec{x}) \geq t(y,\vec{x}))$

(iii) $(\forall z < f_3(\vec{x}))(\neg A(z)) \wedge (f_3(\vec{x}) \leq |s| \supset A(f_3(x))) \wedge f_3(\vec{x}) \leq |s|+1$

Proof: This is actually what we proved in Theorem 9. Note we have defined f_3 so that if $(\forall y \leq |s|)\neg A(y)$ then $f_3(\vec{x})=|s|+1$. □

We are now ready to introduce function and predicate symbols in S_2^1 for handling general sequences. We leave the provability of the necessary uniqueness and existence conditions to the reader.

(a) $b=Substring(a,i,j) \iff b=MSP(LSP(a,j),i)$

So the binary representation of b is that portion of a's binary representation starting with the 2^{j-1} bit and ending with the 2^i bit. For example, if $a=13_{10}=1101_2$ then $Substring(a,0,3)=101_2=5_{10}$ and $Substring(a,1,3)=10_2=2_{10}$.

(b) $Seq(w) \iff (\forall x<|w|)[Even(i) \supset Comma(i,w) \vee Digit(i,w) \leq 1] \wedge (Comma(0,w) \vee w=0)$

So a sequence is any number whose binary representation codes a string of 0's, 1's and commas, provided that the two low order bits code a comma (also, the number 0 codes a sequence). We are requiring that the two lowest order bits code a comma so that we can treat the empty and non-empty sequences uniformly.

(c) $a=Len(w) \iff (\neg Seq(w) \wedge a=0) \vee (Seq(w) \wedge a=(\#i<|w|)Comma(i,w))$

(d) $b=Decode(a) \iff (b=0 \wedge \neg ProtoSeq(a)) \vee (ProtoSeq(a) \wedge b=Proto\beta(1,a))$

$b=Encode(a) \iff PSqSL(b,|a|,1) \wedge a=Proto\beta(1,b)$

The existence condition for *Encode* follows from the remark made in §2.4(o) above.

(e) $$a{=}Start\beta(i,w) \iff (a{=}0\wedge\neg Seq(w))\vee(Seq(w)\wedge \wedge\, a{=}(\mu x{\le}|w|{+}1)[Len(Substring(w,0,x)){=}i\wedge Even(x)])$$

$$b{=}End\beta(i,w) \iff (a{=}0\wedge\neg Seq(w))\vee(Seq(w)\wedge \wedge\, b{=}(\mu x{\le}|w|)[Len(Substring(w,Start\beta(i,w),x{+}2)){\ne}0\wedge Even(x)])$$

$$a{=}\beta(i,w) \iff (i{=}0\wedge a{=}Len(w))\vee \vee(a{=}Decode(Substring(w,Start\beta(i,w),End\beta(i,w)))\wedge i{\ne}0)$$

Note the β function is defined so that $\beta(0,w){=}Len(w)$.

(f) $$Size(w) = \max\{\lfloor\tfrac{1}{2}(End\beta(i,w)\dot{-}Start\beta(i,w))\rfloor : i{\le}Len(w)\}$$

(g) $$a{*}{*}b = b\cdot 2^{|a|+Mod2(|a|)}+a$$

(h) $$a{*}b = a{*}{*}(4\cdot Encode(b)+1)$$

Note that unlike the conventions in Chapter 1, $<a_1,\ldots,a_n>*a_{n+1}$ is $<a_1,\ldots,a_{n+1}>$. Also, from now on, $*$ associates from left to right.

(i) $$Subseq(w,i,j) = Substring(w,End\beta(i\dot{-}1,w),End\beta(j\dot{-}1,w))$$

So $Subseq(w,i,j)$ is the subsequence $<\beta(i,w),\ldots,\beta(j\dot{-}1,w)>$ of w.

(j) $$UniqSeq(w) \iff Seq(w)\wedge Digit(|w|\dot{-}2,w){\ne}0\wedge \wedge\neg(\exists i{\le}|w|)(Digit(i,w){=}0\wedge Comma(i{+}2,w))$$

$UniqSeq(w)$ asserts that w is a sequence and that all entries in w are coded without any extraneous leading zeros. The reason we are interested in $UniqSeq$ is that S^1_2 proves

$$UniqSeq(a)\wedge UniqSeq(b)\wedge(\forall i{\le}Len(a))(\beta(i,a){=}\beta(i,b))\supset a{=}b$$

and

$$Seq(a)\supset(\exists w)(UniqSeq(w)\wedge(\forall i{\le}Len(a))(\beta(i,a){=}\beta(i,w))).$$

(k) $$SqBd(a,b) = (2b{+}1)\#(4\cdot(2\cdot a{+}1)^2)$$

SqBd is useful since

$$S_2^1 \vdash UniqSeq(w) \wedge Len(w) \leq |b|+1 \wedge (\forall i < Len(w))(\beta(Si,w) \leq a) \supset w < SqBd(a,b).$$

2.6. Bootstrapping $\mathbf{T_2^1}$.

Now that we have completed the bootstrapping of S_2^1, we want to bootstrap T_2^1. Fortunately we will need to do much less work to bootstrap T_2^1. Indeed, once we have defined a few simple functions, we will be able to show that T_2^1 proves all the Σ_1^b–*PIND* axioms. Hence $T_2^1 \supseteq S_2^1$ and all the functions defined in the last two sections can be introduced into T_2^1.

We begin by showing that the following functions may be introduced in T_2^1:

(a) $a < b \iff a \leq b \wedge a \neq b$

$c = \max(a,b) \iff (c \geq a \wedge c = b) \vee (c \geq b \wedge c = a)$

$c = \min(a,b) \iff (c \leq a \wedge c = b) \vee (c \leq b \wedge c = a)$

(b) $b = P(a) \iff (a=0 \wedge b=0) \vee Sb = a$

We showed in an example earlier that subtraction is Σ_1^b–definable in T_2^1. So we can define $P(a) = a \dot{-} 1$.

(c) $Power2(a) \iff S(|P(a)|) = |a|$

When we introduced *Power2* as a defined predicate symbol of S_2^1 we showed that S_2^1 can prove many basic properties of the *Power2* predicate. The same comments apply to T_2^1.

(d) $c = Exp(a,b) \iff Power2(c) \wedge |c| = 1 + \min(|b|, a)$

i.e., $Exp(a,b) = 2^{\min(|b|,a)}$.

Let $M(a,b,c)$ be the righthand side of the defining equation for *Exp*. Then,

$$T_2^1 \vdash M(x,y,z) \wedge (x \leq |y| \vee |Sy| = |y|) \supset M(x,Sy,z)$$

and

$$T_2^1 \vdash M(x,y,z) \wedge x > |y| \wedge |Sy| > |y| \supset M(x,Sy,2z).$$

Hence,

$$T_2^1 \vdash (\exists z \le 2y+1) M(x,y,z) \supset (\exists z \le 2\cdot Sy+1) M(x,Sy,z).$$

Since $T_2^1 \vdash M(x,0,1)$, we can use Σ_1^b–*IND* to obtain

$$T_2^1 \vdash (\forall x)(\forall y)(\exists z \le 2y+1) M(x,y,z)$$

which is the existence condition for the definition of *Exp*.

(e) $Decomp(a,b,c,d) \iff |c| \le b \wedge (d\cdot 2^{\min(|a|,b)} + c = a)$

$c = LSP(a,b) \iff (\exists d \le a) Decomp(a,b,c,d)$

$d = MSP(a,b) \iff (\exists c \le a) Decomp(a,b,c,d)$

It is easy to see that *Decomp* may be introduced as a Δ_1^b–defined predicate symbol of T_2^1. Also, the uniqueness conditions for *LSP* and *MSP* follow from the *BASIC* axioms. It will suffice therefore to show that

$$T_2^1 \vdash (\exists x \le a)(\exists y \le a) Decomp(a,b,x,y).$$

But

$$T_2^1 \vdash Decomp(a,b,c,d) \wedge c+1 < 2^{\min(b,|a|)} \supset Decomp(a+1,b,c+1,d)$$

and

$$T_2^1 \vdash Decomp(a,b,c,d) \wedge c+1 \ge 2^{\min(b,|a|)} \supset Decomp(a+1,b,0,d+1).$$

Hence we can use Σ_1^b–*IND* to prove the existence condition.

Definition: When Q and R are theories, we write $Q \vdash R$ to mean that every theorem of R is a theorem of Q.

Theorem 11: Let $i \ge 1$. T_2^i proves the Σ_i^b–*PIND* axioms. Hence $T_2^i \vdash S_2^i$.

Proof: Let A be any Σ_i^b–formula. We want to show that

$$T_2^i \vdash A(0) \wedge (\forall x)(A(\lfloor \tfrac{1}{2}x \rfloor) \supset A(x)) \supset A(a)$$

(where a is a free variable which appears only as indicated.) Let $B(a,u)$ be the formula $A(MSP(a,|a| \dot- u))$. Then

$$T_2^i \vdash A(0) \supset B(a,0)$$

$$T_2^i \vdash (\forall x)(A(\lfloor \tfrac{1}{2}x \rfloor) \supset A(x)) \supset (\forall x)(B(a,x) \supset B(a,Sx)).$$

Now, from Σ_i^b-*LIND* on B, we have

$$T_2^i \vdash B(0) \wedge (\forall x)(B(x) \supset B(Sx)) \supset B(|a|).$$

But,

$$T_2^i \vdash B(|a|) \supset A(a).$$

Q.E.D. □

If we examine the proof of Theorem 11, we note that only Σ_i^b-*LIND* is used, not Σ_i^b-*IND*. Hence what we have proved is:

Theorem 12: Let R_i be the theory S_2^1 plus the Σ_i^b-*LIND* axioms. Then R_i is equivalent to S_2^i.

Proof: $R_i \vdash S_2^i$ is proved by the proof of Theorem 11. Theorem 6 implies that $S_2^i \vdash R_i$.

Q.E.D. □

Theorem 13: Let $i \geq 0$. Then
(a) $S_2^1 + \Sigma_i^b$-*LIND* is equivalent to $S_2^1 + \Sigma_i^b$-*PIND*.
(b) $S_2^1 + \Sigma_i^b$-*IND* is equivalent to $S_2^1 + \Pi_i^b$-*IND*.
(c) $S_2^1 + \Sigma_i^b$-*LIND* is equivalent to $S_2^1 + \Pi_i^b$-*LIND*.
(d) $S_2^1 + \Sigma_i^b$-*PIND* is equivalent to $S_2^1 + \Pi_i^b$-*PIND*.

Proof: The inclusion of S_2^1 means that we can use all of the Σ_1^b-defined function symbols of S_2^1 freely.
(a) By Theorem 12.
(b) One half of this is Theorem 5. The other direction is proved by exactly the same idea of "reversing" the direction of the induction.
(c) This is proved by an argument similar to the proof of (b).
(d) By (a) and (c) it suffices to show that $S_2^1 + \Pi_i^b$-*LIND* is equivalent to $S_2^1 + \Pi_1^b$-*PIND*. $S_2^1 + \Pi_i^b$-*PIND* $\Longrightarrow \Pi_i^b$-*LIND* follows from the proof of Theorem 6 modified so that $A \in \Pi_i^b$ instead of Σ_i^b. Likewise, the proof of Theorem 11 modified so that $A \in \Pi_i^b$ shows that $S_2^1 + \Pi_i^b$-*LIND* $\Longrightarrow \Pi_i^b$-*PIND*.

Q.E.D. □

2.7. Replacement Axioms.

An important property of the natural numbers is the replacement axiom, also called the collection axiom. This axiom is $(\forall x \le a)(\exists y)A \leftrightarrow (\exists t)(\forall x \le a)(\exists y \le t)A$. One of the reasons this axiom is useful is that it shows that unbounded quantifiers may be moved outside the scope of bounded quantifiers. In the classical setting, it is the unbounded quantifiers which are most important and the bounded quantifiers are generally ignored, and the replacement axioms state that the order of bounded and unbounded quantifiers may be exchanged.

In our setting, however, bounded quantifiers are important and the sharply bounded quantifiers are generally ignored. A natural question is whether there is a version of the replacement axiom for our setting. The answer is partly yes, in that bounded quantifiers may be moved outside sharply bounded quantifiers.

Definition: The Σ_i^b-replacement axioms are the formulae of the form

$$(\forall x \le |t|)(\exists y \le s)A(x,y) \leftrightarrow (\exists w \le SqBd(s,t))(\forall x \le |t|)(A(x,\beta(Sx,w)) \wedge \beta(Sx,w) \le s)$$

where s and t are arbitrary terms and A is any Σ_i^b-formula, and other free variables may appear in A.

Theorem 14: Let $i \ge 1$. Then the Σ_i^b-replacement axioms are theorems of S_2^i.

Proof: Let A be any Σ_i^b-formula. Let Y and Z be the formulae

$$Y = (\forall x \le |t|)(\exists y \le s)A(x,y)$$

$$Z(u) = (\exists w \le SqBd(s,t))(\forall x \le |t|)(x \le u \supset A(x,\beta(Sx,w)) \wedge \beta(Sx,w) \le s).$$

We want to show $S_2^i \vdash Y \leftrightarrow Z(|t|)$. Now, $S_2^i \vdash Z(|t|) \supset Y$ is obvious. Also,

$$S_2^i \vdash Y \supset Z(0)$$

and

$$S_2^i \vdash Y \wedge Z(u) \wedge u < |t| \supset Z(Su).$$

Thus, by Σ_i^b-$LIND$, $S_2^i \vdash Y \supset Z(|t|)$.

Q.E.D. □

Definition: The sets $\Sigma_i^b(AS)$ and $\Pi_i^b(AS)$ are defined inductively by:

(1) $\Sigma_0^b(AS)$ is the set of Π_1^b-formulae which are Δ_1^b with respect to the theory S_2^1. Similarly, $\Pi_0^b(AS)$ is the set of Σ_1^b-formulae which are Δ_1^b with respect to the theory S_2^1.

(2) $\Sigma_{i+1}^b(AS)$ is the smallest set satisfying:
(a) $\Sigma_{i+1}^b(AS) \supseteq \Pi_i^b(AS)$ and
(b) If $A \in \Sigma_{i+1}^b(AS)$ then $(\exists x \le t)A$ is in $\Sigma_{i+1}^b(AS)$.

(3) $\Pi_{i+1}^b(AS)$ is the smallest set satisfying:
(a) $\Pi_{i+1}^b(AS) \supseteq \Sigma_i^b(AS)$ and
(b) If $A \in \Pi_{i+1}^b(AS)$ then $(\forall x \le t)A$ is in $\Pi_{i+1}^b(AS)$.

The (AS) means *alternative sense.* Note that Σ_0^b is a proper subset of $\Sigma_0^b(AS)$ and that $\Sigma_{i+1}^b(AS)$ is a proper subset of Σ_{i+1}^b.

Let R_i be the theory S_2^1 plus the Σ_i^b-replacement axioms.

Corollary 15: If A is a Σ_i^b- or a Π_i^b-formula, then there is a $\Sigma_i^b(AS)$- or a $\Pi_i^b(AS)$-formula B (respectively) which is provably equivalent to A in the theory R_i.

Corollary 15 is easily proved by induction on the complexity of A. Note that we are using the fact that the function β is Σ_1^b-definable. Theorem 14 asserts that $S_2^i \vdash R_i$. Although we don't know if the converse is true, we do have the following theorem:

Theorem 16: $R_{i+1} \vdash S_2^i$.

Proof: by induction on i. For $i=1$ it is obvious. So assume $i \ge 2$ and $R_{i+1} \vdash S_2^{i-1}$. By Theorem 13 it suffices to show that R_{i+1} proves every Σ_i^b-*LIND* axiom.

Let A be any Σ_i^b-formula. We want to show

$$R_{i+1} \vdash A(0) \wedge (\forall x)(A(x) \supset A(Sx)) \supset (\forall x)A(|x|).$$

By Corollary 15, there is a $\Sigma_i^b(AS)$-formula B such that $R_i \vdash A(x) \leftrightarrow B(x)$. Let B have the form

$$B(x) = (\exists y_1 \le t_1) \cdots (\exists y_n \le t_n) C(x, y_1, \ldots, y_n)$$

where $C \in \Pi_{i-1}^b$. We assume without loss of generality that the terms t_i do not include the variables $y_1, \ldots, y_n$. Of course x will generally appear in t_i. For notational simplicity we assume that $n=1$ for the rest of the proof and write $t(x)$ instead of $t_1(x)$.

Let D be the formula

$$D(x,y) \;=\; C(x,y) \wedge y \le t(x).$$

Let u be a new variable. Then, by prenex operations,

$$R_{i+1} \vdash (\forall x)(B(x) \supset B(Sx)) \supset (\forall x < |u|)(\exists y \leq t(Sx))(\forall z \leq t(x))(D(x,z) \supset D(Sx,y)).$$

By Σ_{i+1}^b–replacement,

$$R_{i+1} \vdash (\forall x)(B(x) \supset B(Sx)) \supset (\exists w)(\forall x < |u|)(\forall z \leq t(x))(D(x,z) \supset D(Sx,\beta(Sx,w))).$$

Let f be the Σ_1^b–definable function satisfying

$$f(a,w,b) = \begin{cases} b & \text{if } a=0 \\ \beta(a,w) & \text{if } a>0 \end{cases}$$

Thus,

$$R_{i+1} \vdash D(0,b) \wedge (\forall x)(B(x) \supset B(Sx)) \supset (\exists w)(\forall x < |u|)(D(x,f(x,w,b)) \supset D(Sx,\beta(Sx,w))).$$

So,

$$R_{i+1} \vdash B(0) \wedge (\forall x)(B(x) \supset B(Sx)) \supset$$
$$\supset (\exists w)\,[D(0,\beta(1,w)) \wedge (\forall x < |u|)(D(x,\beta(x+1,w)) \supset D(Sx,\beta(x+2,w)))].$$

Since $D \in \Pi_{i-1}^b$, we can use Π_{i-1}^b–*LIND* , to get

$$R_{i+1} \vdash B(0) \wedge (\forall x)(B(x) \supset B(Sx)) \supset (\exists w) D(|u|,\beta(|u|+1,w)).$$

Note that we are justified in using Π_{i-1}^b–*LIND* by our induction hypothesis and by Theorem 13. Finally,

$$R_{i+1} \vdash B(0) \wedge (\forall x)(B(x) \supset B(Sx)) \supset B(|u|).$$

Q.E.D. □

2.8. Minimization Axioms.

We next introduce two new axiom schemas which can be used to axiomatize Bounded Arithmetic.

Definition: Let Ψ be a set of formulae. The Ψ–*MIN* axiom schema consists of the axioms

$$(\exists x)A(x) \supset (\exists x)[A(x) \wedge (\forall y \leq x)(y \neq x \supset \neg A(y))]$$

where A is any formula in Ψ.

The Ψ–*LMIN* axioms are given by the schema

$$(\exists x)A(x) \supset A(0) \vee (\exists x)[A(x) \wedge (\forall y \leq \lfloor \tfrac{1}{2}x \rfloor)(\neg A(y))]$$

where again $A \in \Psi$.

Theorem 17: Let $i \geq 1$. In the theory S_2^1,
(a) Σ_i^b–*MIN* is equivalent to Π_i^b–*IND*, and
(b) Σ_i^b–*LMIN* is equivalent to Π_i^b–*PIND*.

Proof: The proofs of (a) and (b) are almost identical, so we will prove only (b).

First, we show that Σ_i^b–*LMIN* $\Longrightarrow$ Π_i^b–*PIND*. Let $A \in \Pi_i^b$. Then by Σ_i^b–*LMIN*,

$$\neg(\forall x)A(x) \supset \neg A(0) \vee (\exists x)(\neg A(x) \wedge (\forall y \leq \lfloor \tfrac{1}{2}x \rfloor)A(y))$$

and thus

$$\neg(\forall x)A(x) \wedge A(0) \supset (\exists x)(A(\lfloor \tfrac{1}{2}x \rfloor) \wedge \neg A(x))$$

which is what we needed to show.

Secondly, we show Π_i^b–*PIND* $\Longrightarrow$ Σ_i^b–*LMIN*. Let $A(x)$ be a Σ_i^b–formula. Let $B(x)$ be the formula $(\forall y \leq x)(\neg A(y))$. Now, by Π_i^b–*PIND*,

$$\neg B(u) \supset \neg B(0) \vee (\exists x \leq u)(B(\lfloor \tfrac{1}{2}x \rfloor) \wedge \neg B(x))$$

and since $A(u) \supset \neg B(u)$, we have

$$A(u) \supset A(0) \vee (\exists x \leq u)[(\forall z \leq \lfloor \tfrac{1}{2}x \rfloor)(\neg A(z)) \wedge (\exists y \leq x)A(y)].$$

Since the *BASIC* axioms imply $y \leq x \supset \lfloor \tfrac{1}{2}y \rfloor \leq \lfloor \tfrac{1}{2}x \rfloor$ we get

$$A(u) \supset A(0) \vee (\exists y \leq u)(\forall z \leq \lfloor \tfrac{1}{2}y \rfloor)(\neg A(z) \wedge A(y)).$$

The *LMIN* axiom for A is an immediate consequence of this.

Q.E.D. □

By the previous theorem, we can use the minimization axioms instead of induction axioms to axiomatize Bounded Arithmetic. In a more classical setting, Paris and Kirby [21] have studied how minimization axioms can be used to axiomatize fragments of Peano arithmetic. Paris and Kirby have shown that Σ_i^0–*MIN* and Π_i^0–*MIN* are equivalent with respect to a simple open theory P^-. However in Bounded Arithmetic we have a different situation.

Theorem 18: Let $i \geq 1$. $S_2^1 + \Pi_i^b$–*MIN* is equivalent to $S_2^1 + \Sigma_{i+1}^b$–*MIN*.

Proof: Since $\Pi_i^b \subseteq \Sigma_{i+1}^b$, one direction is trivial. We need to show that $\Pi_i^b\text{–}MIN \Longrightarrow \Sigma_{i+1}^b\text{–}MIN$ in the presence of S_2^1. We begin by showing that $S_2^1 + \Pi_i^b$–*MIN* proves the $\Sigma_{i+1}^b(AS)$–*MIN* axioms.

Let $A \in \Sigma_{i+1}^b(AS)$. So $A(x)$ has the form

$$(\exists y_1 \leq t_1(x)) \cdots (\exists y_n \leq t_n(x)) B(x, y_1, \ldots, y_n)$$

where $B \in \Pi_i^b$ (since $i \geq 1$). We can assume without loss of generality that the terms t_i do not include the variables y_j. Let $B^*(x, y_1, \ldots, y_n)$ be the formula

$$B(x, y_1, \ldots, y_n) \wedge y_1 \leq t_1(x) \wedge \cdots \wedge y_n \leq t_n(x).$$

Let $C(w, a)$ be the formula

$$PSqSL(w, a, n+1) \wedge B^*(Proto\beta(n+1, w), Proto\beta(n, w), \ldots, Proto\beta(1, w)).$$

By Theorem 2, C is S_2^1–provably equivalent to a Π_i^b–formula. C asserts that w is a protosequence coding values for x and y_i which witness that $(\exists x) A(x)$ is true. Now,

$$S_2^1 \vdash (\exists x) A(x) \wedge a = \max\{|x|, |t_1(x)|, \ldots, |t_n(x)|\} \supset (\exists w) C(w, a).$$

Since protosequences code entries as fixed length codes,

$$S_2^1 \vdash PSqSL(w, a, n+1) \wedge PSqSL(v, a, n+1) \wedge w \geq v \supset Proto\beta(n+1, w) \geq Proto\beta(n+1, v).$$

So by applying Π_i^b–*MIN*, we get a minimum value for w which satisfies $C(w, a)$ (a is held constant). But now $Proto\beta(n+1, w)$ gives a minimum value for x satisfying $A(x)$. This completes the proof that $S_2^1 + \Pi_i^b$–*MIN* proves the $\Sigma_{i+1}^b(AS)$–*MIN* axioms.

To finish the proof of our current theorem, we must show that $S_2^1 + \Sigma_{i+1}^b(AS)$–*MIN* proves the Σ_{i+1}^b–*MIN* axioms. It will suffice to show that $S_2^1 + \Sigma_{i+1}^b(AS)$–*MIN* proves the Σ_{i+1}^b–replacement axioms, since by Corollary 15 a Σ_{i+1}^b–formula is equivalent to a

$\Sigma_{i+1}^b(AS)$–formula via Σ_{i+1}^b–replacement. The proof of Theorem 17(a) shows that $\Sigma_{i+1}^b(AS)$–$MIN \Longrightarrow \Pi_{i+1}^b(AS)$–$IND$. Also, $S_2^1 + \Pi_{i+1}^b(AS)$–$IND \Longrightarrow \Sigma_{i+1}^b(AS)$–$IND$ can be shown by using the proof of Theorem 5 (this depends on the fact that the defining equation for subtraction ($\dot{-}$) does not contain any sharply bounded quantifiers.) Clearly, $\Sigma_{i+1}^b(AS)$–$IND \Longrightarrow \Sigma_{i+1}^b(AS)$–$LIND$. Hence it suffices to prove the following lemma.

Lemma 19: Let $i \geq 0$. $S_2^1 + \Sigma_{i+1}^b(AS)$–$LIND \Longrightarrow \Sigma_{i+1}^b$–replacement.

Proof: For $i=0$, this lemma is a consequence of Theorem 14. For $i \geq 1$, by Theorems 14 and 13(a) it suffices to show that $S_2^1 + \Sigma_{i+1}^b(AS)$–$LIND$ proves that every Σ_{i+1}^b–formula is equivalent to a $\Sigma_{i+1}^b(AS)$–formula. The proof of this lemma is a more complicated version of the proof of Corollary 15 which we omitted earlier.

Suppose, for the sake of contradiction, that $2 \leq j \leq i+1$ and that j is the least value for which there exists a Σ_j^b–formula which is not provably equivalent to a $\Sigma_j^b(AS)$–formula by $S_2^1 + \Sigma_{i+1}^b(AS)$–$LIND$.

We shall now show that if $B \in \Sigma_j^b$ then B is provably equivalent to a $\Sigma_j^b(AS)$–formula. It suffices to assume that

$$B \;=\; (\forall x \leq |t|)(\exists y \leq s) A(x,y)$$

where $A \in \Pi_{j-1}^b$, as multiple adjacent existential quantifiers in B can be combined by use of the β function and multiple sharply bounded universal quantifiers can be handled by iterating this argument below.

We prove that B is equivalent to the formula $Z(|t|)$ where $Z(u)$ is the formula

$$(\exists w \leq SqBd(s,t))(\forall x \leq |t|)(x \leq u \supset A(x, \beta(Sx,w)) \wedge \beta(Sx,w) \leq s).$$

We use the proof of Theorem 14 to prove this. The crucial point of the proof of Theorem 14 used Σ_j^b–$LIND$ on the formula $Z(u)$. But how can we use $LIND$ on Z? Well, by our choice of j and since $A \in \Pi_{j-1}^b$, $S_2^1 + \Sigma_{i+1}^b(AS)$–$LIND$ proves that Z is equivalent to a $\Sigma_j^b(AS)$–formula. Hence we are justified in using $LIND$ on the formula Z.

This completes the proof of the lemma and of Theorem 18.

Q.E.D. □

(Remark: In the original version of this dissertation we erroneously claimed to have proved Theorems 18 and 20 for $i \geq 0$ instead of $i \geq 1$.)

An important theorem about the minimization axioms is the following.

Theorem 20: Let $i \geq 1$. The Σ_i^b–MIN axioms are theorems of S_2^{i+1}.

Proof: Let $A(x)$ be any Σ_i^b–formula. Let $B(a,b,c)$ be

$$(\forall x\leq|a|)(x<|a|\dot{-}b\supset Bit(x,c)=0)\wedge(\forall y<c)(\neg A(y))\wedge(\exists y<2^{|a|\dot{-}b})A(c+y).$$

Clearly,

$$S_2^1\vdash A(a)\supset B(a,0,0)\vee a=0.$$

So,

$$S_2^1\vdash A(a)\wedge a\neq 0\supset(\exists x\leq a)B(a,0,x)$$

We also claim that

$$S_2^1\vdash A(a)\wedge a\neq 0\wedge b<|a|\wedge(\exists x\leq a)B(a,b,x)\supset(\exists x\leq a)B(a,Sb,x).$$

This is true because

$$S_2^1\vdash b<|a|\wedge B(a,b,c)\wedge(\exists y<2^{|a|\dot{-}(b+1)})A(c+y)\supset B(a,Sb,c)$$

and

$$S_2^1\vdash b<|a|\wedge B(a,b,c)\wedge(\forall y<2^{|a|\dot{-}(b+1)})(\neg A(c+y))\supset B(a,Sb,c+2^{|a|\dot{-}(b+1)}).$$

These last two results follow from the bit manipulation techniques developed while bootstrapping S_2^1. Finally, from the definition of B we have

$$S_2^1\vdash A(a)\wedge B(a,b,c)\supset c\leq a.$$

Putting the above results together proves the claim.

Since B is a Σ_{i+1}^b–formula, we can use Σ_{i+1}^b–*LIND* on the formula $(\exists x\leq a)B(a,b,x)$ to get

$$S_2^1\vdash A(a)\wedge a\neq 0\supset(\exists x\leq a)B(a,|a|,x)$$

From this the Σ_{i+1}^b–*MIN* axiom for A is immediate.

Q.E.D. □

Corollary 21: If $i\geq 0$, $S_2^{i+1}\vdash T_2^i$.

Proof: For $i\geq 1$, this follows from Theorems 20, 17(a) and 13(b). For $i=0$, this is a corollary to the next theorem. □

The next theorem provides a direct proof of the previous corollary; in fact it is somewhat stronger. Furthermore, the proof does not depend on any of the earlier theorems in this section. Recall that the Δ_i^b formulae are those provably equivalent to both a Σ_i^b– and a Π_i^b–formula.

Theorem 22: Let $i\geq 1$. The Δ_i^b–*IND* axioms are theorems of S_2^i. (Δ_i^b means with respect to S_2^i.)

Proof: (according to M. Dowd [8], the case i=1 is independently due to R. Statman)

Let A be a formula such that there are formulae A^Σ in Σ_i^b and A^Π in Π_i^b such that $S_2^i\vdash A\leftrightarrow A^\Sigma$ and $S_2^i\vdash A\leftrightarrow A^\Pi$. Let $B(x,z)$ be the formula

$$(\forall y\leq Sz)(A(x\dot{-}y)\supset A(x))$$

so B is provably equivalent to a Π_i^b–formula. We claim that

$$S_2^i\vdash(\forall x\leq c)B(x,\lfloor\tfrac{1}{2}d\rfloor)\supset(\forall x\leq c)B(x,d)$$

where c and d are new free variables. This is because $A(x\dot{-}y)\supset A(x)$ follows from $A(x\dot{-}y)\supset A(x\dot{-}\lfloor\frac{1}{2}y\rfloor)$ and $A(x\dot{-}\lfloor\frac{1}{2}y\rfloor)\supset A(x)$. So by Π_i^b–*PIND* ,

$$S_2^i\vdash(\forall x\leq c)B(x,0)\supset(\forall x\leq c)B(x,c).$$

But clearly, $(\forall x\leq c)B(x,c)\supset(A(0)\supset A(c))$ and $(\forall x)(A(x)\supset A(Sx))\supset(\forall x\leq c)B(x,0)$ are provable in S_2^1. Hence, S_2^1 proves

$$(\forall x)(A(x)\supset A(Sx))\supset(A(0)\supset A(c))$$

and the desired induction axiom for A follows immediately by a $\forall$–introduction, since c is a free variable which occurs only as indicated in the last formula.

Q.E.D. □

2.9. Summary of Axiomatizations of Bounded Arithmetic.

We briefly summarize some of the results of this chapter.

Theorem 23: For all $i \geq 0$, $T_2^{i+1} \Longrightarrow S_2^{i+1}$ and $S_2^{i+1} \Longrightarrow T_2^i$.

Proof: by Theorem 11 and Corollary 21. □

Theorem 24: Let $i \geq 0$. In the presence of S_2^1, we have the following implications:

(a)
$$\begin{tikzcd}
\Sigma_{i+1}^b\text{-}IND \arrow[r, Leftrightarrow] \arrow[d, Rightarrow] & \Pi_{i+1}^b\text{-}IND \arrow[r, Leftrightarrow] & \Sigma_{i+1}^b\text{-}MIN & \\
\Sigma_{i+1}^b\text{-}PIND \arrow[r, Leftrightarrow] \arrow[d, Leftrightarrow] & \Pi_{i+1}^b\text{-}PIND \arrow[r, Leftrightarrow] & \Sigma_{i+1}^b\text{-}LIND \arrow[r, Leftrightarrow] & \Pi_{i+1}^b\text{-}LIND \\
\Sigma_{i+1}^b\text{-}LMIN \arrow[d, Rightarrow] & & & \\
\Sigma_i^b\text{-}IND & & &
\end{tikzcd}$$

(b) $\Sigma_{i+2}^b\text{-}MIN \iff \Pi_{i+1}^b\text{-}MIN$

(c) Σ_{i+1}^b-replacement $\Longrightarrow$ $\Sigma_i^b\text{-}PIND$ $\Longrightarrow$ Σ_i^b-replacement.

Proof: By Theorems 5, 6, 11, 13, 14, 16, 17, and 18 and Corollary 21. □

Chapter 3

Definability of Polynomial Hierarchy Functions

The previous chapter investigated several different ways to axiomatize Bounded Arithmetic. We will now be concerned exclusively with the fragments of Bounded Arithmetic axiomatized by *PIND* axioms, that is to say, with the theories S_2^i.

It turns out that using *PIND* is a very natural way to define Bounded Arithmetic. Indeed, there is a very close relationship between the theories S_2^i and the polynomial hierarchy. We discuss part of the relationship in this chapter. The rest is established in Chapter 5.

In Chapter 1, we defined a polynomial hierarchy of both predicates and functions. The classes of predicates were Σ_k^p, Π_k^p and Δ_k^p, where Σ_1^p is NP and Δ_1^p is P. In Chapter 1, we considered a predicate to be a function with range $\{0,1\}$ with the value 0 denoting "false" and 1 denoting "true". We will no longer follow this convention; instead, we think of a predicate in the usual sense as a property of natural numbers.

The classes $\square_k^p$ formed the polynomial hierarchy of functions. The functions in $\square_k^p$ are the functions which are computable in polynomial time by a Turing machine (for computer scientists, a transducer) with an oracle for a predicate in Σ_{k-1}^p. For example, $\square_1^p$ is the set of functions computable in polynomial time.

Theorem 1: Let $k\geq 1$. Let f be an m-ary $\square_k^p$–function. Let $t(\vec{x})$ be a term (in the language of Bounded Arithmetic) so that for all $\vec{x}\in\mathbf{N}^m$, $f(\vec{x})\leq t(\vec{x})$. Then there is a Σ_k^b–formula A such that

(1) $S_2^k \vdash (\forall\vec{x})(\exists y\leq t)A(\vec{x},y)$

(2) $S_2^k \vdash (\forall\vec{x})(\forall y)(\forall z)(A(\vec{x},y)\wedge A(\vec{x},z)\supset y=z)$

(3) For all $\vec{x}\in\mathbf{N}^m$, $A(\vec{x},f(\vec{x}))$ is true.

Theorem 1 says that the theory S_2^k can Σ_k^b–define all of the functions which are polynomial time computable relative to the Σ_{k-1}^p predicates. We will prove the converse of this in Chapter 5.

Proof: First we examine the condition of the term t bounding f. Suppose that (1)-(3) hold and that $s(\vec{x})$ is another term such that for all $\vec{x}\in\mathbf{N}^m$, $f(\vec{x})\leq s(\vec{x})$. Let B be the formula

$$B(\vec{x},y) \;=\; (\exists z\leq t)(A(\vec{x},z)\wedge y=\min(s,z)).$$

Then,

$$S_2^k \vdash (\forall \vec{x})(\exists y \leq s) B(\vec{x}, y)$$

$$S_2^k \vdash (\forall \vec{x})(\forall y)(\forall w)(B(\vec{x}, y) \wedge B(\vec{x}, w) \supset y = w)$$

and for all $\vec{x} \in \mathbf{N}^m$, $B(\vec{x}, f(\vec{x}))$ is true.

Thus it will suffice to prove that if $f \in \square_k^p$ then there exists some term t such that (1)-(3) hold. We prove this by induction on the complexity of the definition of f. To begin the induction argument we consider functions f in the set B defined in Chapter 1. In the induction step we will consider separate cases for f defined by composition, limited iteration, or bounded quantification from previously defined functions.

Case (1): Suppose $f \in B$. Clearly f can be Σ_1^b–defined by S_2^1.

Case (2): Suppose f is defined by composition as $f(\vec{x}) = g(h_1(\vec{x}), \ldots, h_n(\vec{x}))$ where g, $h_1, \ldots, h_n$ are functions in $\square_k^p$ and that S_2^k can Σ_k^b–define g, $h_1, \ldots, h_n$ with the formulae

$$S_2^k \vdash (\exists! z \leq s(y_1, \ldots, y_n)) A_g(y_1, \ldots, y_n, z)$$

$$S_2^k \vdash (\exists! z \leq r_i(\vec{x})) A_i(\vec{x}, z)$$

respectively. Let $A(\vec{x}, z)$ be the formula

$$(\exists y_1 \leq r_1(\vec{x})) \cdots (\exists y_n \leq r_n(\vec{x}))(A_g(y_1, \ldots, y_n, z) \wedge A_1(\vec{x}, y_1) \wedge \cdots \wedge A_n(\vec{x}, y_n)).$$

Then $A \in \Sigma_k^b$ and for all $x_i \in \mathbf{N}$, $A(\vec{x}, f(\vec{x}))$ is true. Let $t(\vec{x})$ be the term $s(r_1, \ldots, r_n)$. Then conditions (1)-(3) of the theorem hold.

Case (3): Suppose f is defined from g by bounded existential quantification (i.e. $PB\exists$). That is to say,

$$f(\vec{x}) = \begin{cases} 1 & \text{if } (\exists u \leq s)(g(u, \vec{x}) \neq 0) \\ 0 & \text{otherwise} \end{cases}$$

Suppose also that g is Σ_{k-1}^b–definable by S_2^{k-1} with the defining condition

$$S_2^{k-1} \vdash (\forall u)(\forall \vec{x})(\exists! y \leq r(u, \vec{x})) A_g(u, \vec{x}, y)$$

where A_g is a Σ_{k-1}^b–formula. Let $A(\vec{x}, z)$ be

$$[z = 1 \wedge (\exists u \leq s)(\exists y \leq r)(y \neq 0 \wedge A_g(u, \vec{x}, y))] \vee [z = 0 \wedge (\forall u \leq s)(\forall y \leq r)(y \neq 0 \supset \neg A_g(u, \vec{x}, y))].$$

Then for all values of $\vec{x}$, $A(\vec{x},f(\vec{x}))$ is true. Also A is clearly Δ_k^b. Let $t(\vec{x})$ be the constant term 1. Then conditions (1)-(3) are satisfied.

Case (4): Suppose f is defined by limited iteration from g and h with time bound p and space bound q. Also suppose g and h are Σ_k^b–defined by S_2^k by the defining conditions

$$S_2^k \vdash (\forall \vec{x})(\exists ! z \leq s) A_g(\vec{x},z)$$

$$S_2^k \vdash (\forall \vec{x})(\forall u)(\forall v)(\exists ! z \leq r(\vec{x},u,v)) A_h(\vec{x},u,v,z).$$

Define $B(w,u)$ to be the formula

$$Seq(w) \wedge Len(w) = u+1 \wedge \beta(1,w) = \min(2^{q(|\vec{x}|)}, g(\vec{x})) \wedge$$
$$\wedge (\forall i < u)(\beta(i+2,w) = \min(2^{q(|\vec{x}|)}, h(\vec{x},i,\beta(i+1,w)))).$$

So $B(w,u)$ asserts that w codes the first u steps of the computation of f from g and h, where we are adopting the convention that if the next iteration step would violate the space bound q, then the computation of f is aborted. It is not hard to see that

$$S_2^k \vdash (\exists w \leq SqBd(2^{q(|\vec{x}|)}, \lfloor \tfrac{1}{2} a \rfloor)) B(w, |\lfloor \tfrac{1}{2} a \rfloor|) \supset (\exists w \leq SqBd(2^{q(|\vec{x}|)}, a)) B(w, |a|).$$

Note that B is a Σ_k^b–formula since the quantifier $(\forall i < u)$ is equivalent to a sharply bounded quantifier. So by Σ_k^b–*PIND* ,

$$S_2^k \vdash (\exists w \leq SqBd(2^{q(|\vec{x}|)}, 2^{p(|\vec{x}|)})) B(w, p(|\vec{x}|)).$$

Also, S_2^k proves that this sequence w is unique by the use of Σ_k^b–*LIND* on the length of w. So let $A(\vec{x},y)$ be the formula

$$(\exists w \leq SqBd(2^{q(|\vec{x}|)}, 2^{p(|\vec{x}|)}))(B(w, p(|\vec{x}|)) \wedge y = \beta(p(|x|)+1, w)).$$

Let $t(\vec{x})$ be $2^{q(|\vec{x}|)}$. Then conditions (1)-(3) hold.

Q.E.D. □

We have a similar theorem regarding the definability of Δ_k^p predicates in S_2^k.

Theorem 2: Let $k \geq 1$. Let Q be an m-ary Δ_k^p predicate. Then there are formulae A and B in Σ_k^b and Π_k^b, respectively, so that

(1) $S_2^k \vdash (\forall \vec{x})(A(\vec{x}) \leftrightarrow B(\vec{x}))$
(2) For all $\vec{n} \in \mathbf{N}^m$, $A(\vec{n}) \Longleftrightarrow B(\vec{n}) \Longleftrightarrow Q(\vec{n})$.

Proof: Let f be the $\square_k^p$ function defined by

$$f(\vec{x}) = \begin{cases} 1 & \text{if } Q(\vec{x}) \\ 0 & \text{if } \neg Q(\vec{x}) \end{cases}$$

Let $t(\vec{x})$ be the constant term 1. Let A_f be a Σ_k^b–formula satisfying (1)-(3) of Theorem 1. Define A and B to be

$$A(\vec{x}) = A_f(x,1)$$

$$B(\vec{x}) = \neg A_f(x,0).$$

Then $A \in \Sigma_k^b$ and $B \in \Pi_k^b$ and the theorem is proved.

Q.E.D. □

If we consider the case $k=1$, we get

Corollary 3: Every polynomial time computable function and polynomial time computable predicate can be introduced in S_2^i with a defined function or predicate symbol and used freely in induction formulae (if $i \geq 1$).

Proof: By Theorems 1 and 2 above and Theorems 2.2 and 2.4. □

Chapter 4

First-Order Natural Deduction Systems

This chapter introduces the use of natural deduction systems for first-order Bounded Arithmetic. Up to now we have not been specific about the syntax for our framework of first-order logic; but in order to obtain further results we shall have to make a precise definition of our first-order syntax and rules of deduction. The system we adopt is a modified version of Gentzen's natural deduction calculus LK [13]. An excellent reference for this system is the first half of Takeuti [28]. Several of our proofs will refer to details of proofs in Takeuti [28].

Natural deduction systems provide a very elegant framework for proof-theoretic arguments; they are especially advantageous for proofs which utilize Gentzen's cut elimination theorem.

4.1. Syntax and Rules of Natural Deduction.

Natural deduction uses the following types of symbols:

(1) *Constants*; for example, 0.
(2) *Relations*; for example, $\leq$ and $=$.
(3) *Functions*; for example, S,$+$,$\cdot$,$\#$,$\lfloor\frac{1}{2}x\rfloor$, and $|x|$.
(4) *Free variables*; denoted by $a,b,c,\ldots$
(5) *Bound variables*; denoted by $z,y,x,\ldots$
(6) *Propositional connectives*; $\wedge$,$\vee$,$\supset$, and $\neg$.
(7) *Bounded quantifiers*; $\forall\leq$ and $\exists\leq$.
(8) *Unbounded quantifiers*; $\forall$ and $\exists$.
(9) *Parentheses*.
(10) *Sequent connective*; $\longrightarrow$
(11) *Comma*.

Terms are built up from constants, free variables and functions. *Formulae* are defined as usual. An *atomic formula* is a formula which contains no quantifiers or propositional connectives. An *open* formula is one which contains no quantifiers. A term or formula is *closed* iff it contains no free variables.

A series of formulae separated by commas is called a *cedent*. If Γ and Δ are cedents then $\Gamma \longrightarrow \Delta$ is a *sequent*. The *antecedent* and *succedent* of $\Gamma \longrightarrow \Delta$ are Γ and Δ respectively. The intended meaning of $\Gamma \longrightarrow \Delta$ is that the conjunction of the formulae in Γ implies the disjunction of the formulae in Δ. Although their meanings are similar, $\supset$ and $\longrightarrow$ have very different syntactic roles.

It should be noted there is a distinction between bound and free variables. The set of variables which may appear free in a formula is disjoint from the set of variables which may appear bound in a formula. This is different from the usual conventions of first-order logic, but it does make the syntax more elegant. We use $a,b,c,\ldots$ and $z,y,x,\ldots$ both as variables and as metavariables.

An *inference* is the deduction of a sequent from a set of sequents. An inference is denoted pictorially by

$$\frac{B}{A} \qquad \text{or} \qquad \frac{B \quad C}{A}$$

which means that A is deduced from B or from B and C (each of A, B and C is a sequent).

The rules of natural deduction are listed below. Γ, Π, Λ and Δ are used to denote (parts of) cedents, A and B are arbitrary formulae and s and t are arbitrary terms.

(1) (Weak:left)

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

(2) (Weak:right)

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

(3) (Contraction:left)

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

(4) (Contraction:right)

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

(5) (Exchange:left)

$$\frac{\Gamma, A, B, \Pi \longrightarrow \Delta}{\Gamma, B, A, \Pi \longrightarrow \Delta}$$

(6) (Exchange:right)

$$\frac{\Gamma \longrightarrow \Delta, A, B, \Pi}{\Gamma \longrightarrow \Delta, B, A, \Pi}$$

(7) ($\neg$:left)

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

(8) ($\neg$:right)

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

(9) ($\wedge$:left)

$$\frac{A, \Gamma \longrightarrow \Delta}{A \wedge B, \Gamma \longrightarrow \Delta}$$

and

$$\frac{A, \Gamma \longrightarrow \Delta}{B \wedge A, \Gamma \longrightarrow \Delta}$$

(10) ($\wedge$:right)

$$\frac{\Gamma \longrightarrow \Delta, A \qquad \Gamma \longrightarrow \Delta, B}{\Gamma \longrightarrow \Delta, A \wedge B}$$

(11) ($\vee$:left)

$$\frac{A, \Gamma \longrightarrow \Delta \qquad B, \Gamma \longrightarrow \Delta}{A \vee B, \Gamma \longrightarrow \Delta}$$

(12) ($\vee$:right)

$$\frac{\Gamma \longrightarrow \Delta, A}{\Gamma \longrightarrow \Delta, A \vee B}$$

and

$$\frac{\Gamma \longrightarrow \Delta, A}{\Gamma \longrightarrow \Delta, B \vee A}$$

(13) ($\supset$:left)

$$\frac{\Gamma \longrightarrow \Delta, A \qquad B, \Pi \longrightarrow \Lambda}{A \supset B, \Gamma, \Pi \longrightarrow \Delta, \Lambda}$$

(14) ($\supset$:right)

$$\frac{A, \Gamma \longrightarrow \Delta, B}{\Gamma \longrightarrow \Delta, A \supset B}$$

(15) ($\forall$:left)

$$\frac{A(t), \Gamma \longrightarrow \Delta}{(\forall x)A(x), \Gamma \longrightarrow \Delta}$$

(16) ($\forall$:right)

$$\frac{\Gamma \longrightarrow \Delta, A(a)}{\Gamma \longrightarrow \Delta, (\forall x)A(x)}$$

where a is a free variable which may not appear in the lower sequent of the inference.

(17) ($\exists$:left)

$$\frac{A(a), \Gamma \longrightarrow \Delta}{(\exists x)A(x), \Gamma \longrightarrow \Delta}$$

where a is a free variable which may not appear in the lower sequent of the inference.

(18) ($\exists$:right)

$$\frac{\Gamma \longrightarrow \Delta, A(t)}{\Gamma \longrightarrow \Delta, (\exists x)A(x)}$$

(19) ($\forall\leq$:left)

$$\frac{A(t), \Gamma \longrightarrow \Delta}{t \leq s, (\forall x \leq s)A(x), \Gamma \longrightarrow \Delta}$$

(20) ($\forall\leq$:right)

$$\frac{a \leq t, \Gamma \longrightarrow \Delta, A(a)}{\Gamma \longrightarrow \Delta, (\forall x \leq t)A(x)}$$

where a is a free variable which may not appear in the lower sequent of the inference.

(21) ($\exists\leq$:left)

$$\frac{a\leq t, A(a), \Gamma \longrightarrow \Delta}{(\exists x\leq t)A(x), \Gamma \longrightarrow \Delta}$$

where a is a free variable which may not appear in the lower sequent of the inference.

(22) ($\exists\leq$:right)

$$\frac{\Gamma \longrightarrow \Delta, A(t)}{t\leq s, \Gamma \longrightarrow \Delta, (\exists x\leq s)A(x)}$$

(23) (Cut)

$$\frac{\Gamma \longrightarrow \Delta, A \qquad A, \Pi \longrightarrow \Lambda}{\Gamma, \Pi \longrightarrow \Delta, \Lambda}$$

The inferences (1)-(6) are called *structural inferences.* Rules (7)-(22) are the *logical inferences*: (7)-(14) are the *propositional inferences* and (15)-(22) are the *quantifier inferences.* The formula A in the cut inference is called the *cut formula.* The variable a in inferences (16), (17), (20) and (21) is the *eigenvariable* of the inference. The eigenvariable of an inference must appear only as indicated, or equivalently, must not appear in the conclusion of the inference.

In inferences (7)-(22), the lower sequent contains a newly formed formula which did not appear in the upper sequent. This new formula is called the *principal formula* of the inference. The principal formula of an inference is always formed by using one or more formulae from the upper sequent(s) and by using either a logical symbol or a quantifier. The formula(e) in the upper sequent(s) from which the principal formula is constructed is (are) called the *auxiliary formula(e).* For example, $\neg A$ and $(\exists x\leq s)A(x)$ are the principal formulae of inferences (7) and (22) respectively and their auxiliary formulae are A and $A(t)$ respectively.

A *logical axiom* is a sequent of the form $A \longrightarrow A$ where A must be an atomic formula. An *equality axiom* is a sequent of the form $\longrightarrow t_1 = t_1$,

$$t_1=s_1, \ldots, t_n=s_n \longrightarrow f(t_1, \ldots, t_n)=f(s_1, \ldots, s_n),$$

or

$$t_1=s_1, \ldots, t_n=s_n, p(t_1, \ldots, t_n) \longrightarrow p(s_1, \ldots, s_n)$$

where the t_i's and s_i's are arbitrary terms and f or p is any n-ary function or predicate symbol.

A *proof* is a tree of sequents written so that the root of the tree is at the bottom. The leaves of the tree are called *initial sequents* and must be either equality axioms or logical axioms. Every other sequent in the tree together with the sequents immediately above it must form a valid inference. The root of the tree is called the *endsequent* and it is the formula proved by the proof.

Definition: The natural deduction described above is called *LKB*. (Gentzen's original system *LK* was defined similarly to *LKB* except without equality axioms and without bounded quantifiers.)

Definition: A *bounded formula* is one which contains no unbounded quantifiers. A *bounded sequent* is a sequent which contains only bounded formulae. A *bounded proof* is a proof which contains only bounded sequents.

Proposition 1: *LKB* is consistent, sound and complete.

Proof: The soundness and consistency are obvious. We know that *LK* is complete so it will suffice to show that all properties of bounded quantifiers are theorems of *LKB*.

It is easy to show that for all formulae A, *LKB* proves $A \longrightarrow A$. So consider the following two *LKB*-proofs:

$$\dfrac{\dfrac{\dfrac{A(a) \longrightarrow A(a)}{a \leq t, (\forall x \leq t)A(x) \longrightarrow A(a)}}{(\forall x \leq t)A(x) \longrightarrow a \leq t \supset A(a)}}{(\forall x \leq t)A(x) \longrightarrow (\forall x)(x \leq t \supset A(x))}$$

and

$$\dfrac{\dfrac{\dfrac{a \leq t \longrightarrow a \leq t \qquad A(a) \longrightarrow A(a)}{a \leq t, a \leq t \supset A(a) \longrightarrow A(a)}}{a \leq t, (\forall x)(x \leq t \supset A(x)) \longrightarrow A(a)}}{(\forall x)(x \leq t \supset A(x)) \longrightarrow (\forall x \leq t)A(x)}$$

So *LKB* proves $(\forall x)(x \leq t \supset A(x)) \leftrightarrow (\forall x \leq t)A(x)$. By similar proofs, *LKB* proves that $(\exists x \leq t)A(x)$ is equivalent to $(\exists x)(x \leq t \wedge A(x))$. But now since *LK* is complete, so is *LKB*.

Q.E.D. □

4.2. Bounded Arithmetic.

We next define how systems of Bounded Arithmetic are handled by natural deduction. We must specify how axioms are treated and we must define additional rules of inference.

Definition: The *induction inferences* are:

(1) Σ_i^b-*IND* inference.

$$\frac{\Gamma, A(a) \longrightarrow A(Sa), \Delta}{\Gamma, A(0) \longrightarrow A(t), \Delta}$$

where A is any Σ_i^b-formula, t is any term and a is the *eigenvariable* and must not appear in the lower sequent.

(2) Σ_i^b-*PIND* inference.

$$\frac{\Gamma, A(\lfloor \frac{1}{2}a \rfloor) \longrightarrow A(a), \Delta}{\Gamma, A(0) \longrightarrow A(t), \Delta}$$

with the same provisos as above.

(3) Σ_i^b-*LIND* inference.

$$\frac{\Gamma, A(a) \longrightarrow A(Sa), \Delta}{\Gamma, A(0) \longrightarrow A(|t|), \Delta}$$

where, again, the same provisos apply.

If Ψ is any set of formulae, we define the Ψ-*IND* , Ψ-*PIND* and Ψ-*LIND* inference rules in the same manner.

Definition: Let $A(a_1, \ldots, a_k)$ be a formula with all of A's free variables as indicated. We say B is a *substitution instance* of A iff $B = A(t_1, \ldots, t_k)$ for some terms $t_1, \ldots, t_k$.

Definition: When working in a theory with axioms, we enlarge the notion of proof to allow initial sequents of the form $\longrightarrow A$ where A is any substitution instance of an axiom.

Definition:

(a) S_2^i is the natural deduction theory with the *BASIC* axioms and the Σ_i^b-*PIND* induction inferences.

(b) T_2^i is the natural deduction theory with the *BASIC* axioms and the Σ_i^b-*IND* induction inferences.

Theorem 2: ($i \geq 0$). The Σ_i^b-*IND* (respectively, Σ_i^b-*PIND* , Σ_i^b-*LIND*) rule is equivalent to the Σ_i^b-*IND* (respectively, Σ_i^b-*PIND* , Σ_i^b-*LIND*) axioms. Hence the new definitions of S_2^i and T_2^i agree with the definitions given earlier in Chapter 2.

Proof: It suffices to show that the induction axioms are consequences of the corresponding induction rule (the converse is obvious). We show that the Σ_i^b-*IND* rule can derive the Σ_i^b-*IND* axiom and leave the other cases to the reader.

Let A be any Σ_i^b–formula, and let a and b be any free variables not appearing in A. Then we can derive the *IND* axiom for A by:

$$
\begin{array}{c}
A(a) \longrightarrow A(a) \qquad\qquad A(Sa) \longrightarrow A(Sa) \\
\hline
A(a) \supset A(Sa), A(a) \longrightarrow A(Sa) \\
\hline
(\forall x)(A(x) \supset A(Sx)), A(a) \longrightarrow A(Sa) \\
\hline
(\forall x)(A(x) \supset A(Sx)), A(0) \longrightarrow A(b) \\
\hline
(\forall x)(A(x) \supset A(Sx)), A(0) \longrightarrow (\forall x)A(x) \\
\hline
A(0) \wedge (\forall x)(A(x) \supset A(Sx)), A(0) \longrightarrow (\forall x)A(x) \\
\hline
A(0), A(0) \wedge (\forall x)(A(x) \supset A(Sx)) \longrightarrow (\forall x)A(x) \\
\hline
A(0) \wedge (\forall x)(A(x) \supset A(Sx)), A(0) \wedge (\forall x)(A(x) \supset A(Sx)) \longrightarrow (\forall x)A(x) \\
\hline
A(0) \wedge (\forall x)(A(x) \supset A(Sx)) \longrightarrow (\forall x)A(x) \\
\hline
\longrightarrow A(0) \wedge (\forall x)(A(x) \supset A(Sx)) \supset (\forall x)A(x)
\end{array}
$$

Q.E.D. □

As the above proof shows, natural deduction proofs often can be quite awkward to write out in complete detail. Generally, we shall find it easier to argue informally when we wish to show that a statement is provable.

However, the advantage of natural deduction is that it provides an elegant framework for proof by induction on the complexity of proofs. Generally speaking, natural deduction systems are not a good system with which to prove a theorem; but they are very good for showing that certain things are not provable.

One extremely useful property of natural deduction systems is that proofs can always be put in a normal form. The most important normal form is Gentzen's Hauptsatz, the cut-elimination theorem, which is discussed in the next section.

4.3. Cut Elimination.

The cut elimination theorem is the most fundamental property of natural deduction systems. The cut elimination theorem was first proved by Gentzen [13] and is sometimes referred to as Gentzen's Hauptsatz.

Before we can state the cut elimination theorem in its most general form, we need some more definitions:

Definition: Suppose C is a formula which appears in a given sequent in a proof. The *successor* of C is a formula in the sequent directly below the sequent C appears in. The successor of C is defined according to the following cases:

(1) If C is in the endsequent of the proof or if C is the cut formula of a cut inference, then C has no successor.

(2) If C is the auxiliary formula of an inference, then the principal formula of the inference is the successor of C.

(3) If C is one of the formulae A or B in an exchange inference, the successor of C is the formula denoted by the same letter in the lower sequent of the inference.

(4) If C is the k-th formula in a sub-cedent Γ, Δ, Π or Λ of the upper sequent of an inference, then the successor of C is the k-th formula in the corresponding sub-cedent of the lower sequent of the inference.

(5) If C is the auxiliary formula on the right or left side of an induction inference, then the successor of C is the principal formula on the right or left side respectively.

Definition: Let C and D be occurrences of formulae appearing in a proof. Then C is an *ancestor* of D if there are occurrences $C_1, \ldots, C_n$ of formulae in the proof such that C_1 is C, each C_{i+1} is the successor of C_i and D is the successor of C_n.

We say that C is the ***direct ancestor*** of D iff C is an ancestor of D and C and D are occurrences of the same formula. This means that in the sequence of successors linking C to D, the formulae are never modified by an inference.

If C is an ancestor of D, then we call D a ***descendant*** of C. If C is a direct ancestor of D then D is a ***direct descendant*** of C.

Definition: A formula C appearing in a proof is *free* iff it is not the case that C has a direct ancestor which either is a principal formula of an induction inference or is in an initial sequent.

A cut inference is *free* iff both of the cut formulae in the upper sequents are free.

Remark: We have defined "free cut" somewhat differently from the way Takeuti [28] does. However, the effect of our definition is the same since we required the logical axioms to be atomic. The advantage of our definition is that it allows us to discuss theories which have non-logical axioms which are not open. We shall discuss such theories briefly in Chapter 8.

We are now ready to state the cut elimination theorem:

Theorem 3: (Gentzen) Suppose $\Gamma \longrightarrow \Delta$ is provable in S_2^i or T_2^i by a proof P. Then there is a proof P^* of $\Gamma \longrightarrow \Delta$ in the same theory such that P^* does not have any free cuts. Furthermore each principal formula of an induction inference in P^* is a substitution instance of a principal formula of an induction inference in P.

Proof: This is proved by exactly the same proof as in Takeuti [28], pp. 22-29, 111-112. All that is needed is to add additional cases for the bounded quantifier inferences. This is straightforward and we omit it. □

Corollary 4: (Gentzen) Suppose $\Gamma \longrightarrow \Delta$ is provable in *LKB*. Then $\Gamma \longrightarrow \Delta$ is provable by a proof P such that every cut formula in P is atomic.

Definition: A proof is *cut free* iff no cut inferences appear in the proof. A proof is *free cut free* iff it has no free cuts.

The proof of Theorem 3 is constructive and gives an effective method of finding P^* from P. In fact, the algorithm which accepts P as input and constructs P^* is primitive recursive. However, it is not elementary recursive.

Corollary 5: Let $i \geq 0$. Let Γ and Δ be cedents of Σ_i^b– and Π_i^b–formulae and suppose $\Gamma \longrightarrow \Delta$ is provable in S_2^i or T_2^i. Then there is a proof P of $\Gamma \longrightarrow \Delta$ in S_2^i or T_2^i (respectively) such that every formula in P is in $\Sigma_i^b \cup \Pi_i^b$.

Proof: We pick P to be a free cut free proof of $\Gamma \longrightarrow \Delta$. Suppose C is a formula in P and that $C \notin \Sigma_i^b \cup \Pi_i^b$. Then C can not have been either the principal formula of an induction inference or a direct descendant of a formula in an initial segment. Hence C is free and all of the descendants of C must be free. Since P is free cut free, some descendant of C must appear in the endsequent. However no descendant of C can be in $\Sigma_i^b \cup \Pi_i^b$ and this contradicts the hypotheses of the theorem. Thus all formulae in P must be in $\Sigma_i^b \cup \Pi_i^b$. □

Definition: A cut inference is *inessential* iff its cut formula is atomic.

We shall sometimes use an inessential cut in the construction of a free cut free proof. This is always permissible since the cut formula is atomic and any atomic formula in a proof must be introduced either by an axiom or by a (Weak:left) or a (Weak:right) inference. In the first case the inessential cut is a free cut. In the second case the inessential cut is superfluous in that the proof can be simplified by removing the inessential cut; this is done by deleting the Weak inferences which introduced the cut formula and then replacing the inessential cut by Weak inferences.

Hence we can, without loss of generality, allow arbitrary inessential cuts to appear in free cut free proofs.

4.4. Further Normal Forms for Proofs.

We define some more syntactic properties of proofs.

Definition: Let P be a proof with endsequent $\Gamma \longrightarrow \Delta$. The free variables in $\Gamma \longrightarrow \Delta$ are called the *parameter variables* of P.

We say that P is in *weak free variable normal form* iff for each free variable a in P there is an *elimination inference* such that

(1) a is in the upper sequent(s) of its elimination inference,

(2) a appears in P only above its elimination inference, and

(3) if a appears in a sequent S of P, then a appears in every sequent between S and a's elimination inference,

with the exception that if a is a parameter variable, then we think of the elimination inference for a as being an imaginary inference directly below the endsequent of P.

An alternative, equivalent definition is that P is in *weak free variable normal form* iff for each free variable a in P, the inferences of P which contain a in an upper sequent form a connected subtree of P.

Proposition 6: Let P be a proof in weak free variable normal form and let a be a free variable in P which is not a parameter variable. Then the elimination inference of a must be a ($\forall$:right), ($\forall\leq$:right), ($\exists$:left), ($\exists\leq$:left), ($\forall$:left), ($\exists$:right), or *Cut* inference.

Proof: This is immediate from the syntax of the inferences for Bounded Arithmetic. □

In fact, we can further require that the elimination inference is not a ($\forall$:left), ($\exists$:right), or *Cut* inference:

Proposition 7: Let P be a proof in weak free variable normal form. Suppose a is a free variable in P and the elimination inference for a is a *Cut*, ($\forall$:left) or ($\exists$:right) inference. Then if we replace every occurrence of the free variable a in P by the constant symbol 0 (zero), we still have a valid proof of the same endsequent.

Proof: Examination of the syntax of the inference rules shows that when we carry out the replacement of a by 0, the altered proof is still a valid proof. □

Definition: A proof P is in *free variable normal form* iff P is in weak free variable normal form and for every free variable a appearing in P, the elimination inference for a is not a *Cut*, ($\forall$:left) or ($\exists$:right) inference.

Proposition 8:

(a) Suppose P is a proof of $\Gamma \longrightarrow \Delta$. Then there is a proof P^* of $\Gamma \longrightarrow \Delta$ such that P^* is in free variable normal form.

(a) Suppose P is a proof of $\Gamma \longrightarrow \Delta$. Then there is a proof P^* of $\Gamma \longrightarrow \Delta$ such that P^* is in free variable normal form and P^* is free cut free.

Proof:

(a) P can be transformed to the desired P^* by renaming free variables and using Proposition 7.

(b) First use the cut elimination theorem to obtain a free cut free proof Q of $\Gamma \longrightarrow \Delta$. Then obtain P^* by renaming free variables and using Proposition 7.

Q.E.D. □

4.5. Restricting by Parameter Variables.

The results of this section are somewhat technical in nature. They will be used only in the two sections of Chapter 4 immediately following.

Definition: Let P be a proof. We say that an induction inference in P is *restricted by parameter variables* iff it has the form

$$\frac{\Gamma, A(\lfloor \tfrac{1}{2}a \rfloor) \longrightarrow A(a), \Delta}{\Gamma, A(0) \longrightarrow A(t), \Delta}$$

or

$$\frac{\Gamma, A(a) \longrightarrow A(Sa), \Delta}{\Gamma, A(0) \longrightarrow A(t), \Delta}$$

where the only free variables in the term t are parameter variables of P.

We say P is *restricted by parameter variables* iff every induction inference in P is restricted by parameter variables.

Theorem 9: Let $\Gamma \longrightarrow \Delta$ be a bounded sequent which is provable in one of the theories S_2 or T_2. Then there is a bounded proof of $\Gamma \longrightarrow \Delta$ in the same theory which has no free cuts, is in free variable normal form and is restricted by parameter variables.

Before proving Theorem 9, we introduce a new metafunction σ which lets the proof apply to slightly more general theories. As a bonus, the use of σ may make the proof somewhat easier to understand.

Let R be any theory of arithmetic. We define a metafunction σ which maps terms of the language of R to terms. Suppose $t_1, \ldots, t_k$ are terms with variables $a_1, \ldots, a_s$. Then $\sigma[t_1, \ldots, t_k]$ is a term with the same variables. Furthermore, if $1 \le i \le k$,

$$b_1 \le a_1, \ldots, b_s \le a_s \longrightarrow t_i(b_1, \ldots, b_s) \le \sigma[t_1, \ldots, t_k](a_1, \ldots, a_s)$$

must be provable from the axioms of R **without** the use of any induction inferences.

Obviously the metafunction σ depends on the theory R, and indeed, for a given theory R there are many σ's satisfying the above conditions. The exact choice for σ is not too important, but σ should be as simple and as constructive as possible.

If R is one of the theories S_2 or T_2, we have a particularly simple definition for σ. Define

$$\sigma[t] = t$$

$$\sigma[t_1, \ldots, t_k] = t_1 + \cdots + t_k.$$

This definition works since each function symbol of Bounded Arithmetic is nondecreasing in each of its variables.

If we enlarge S_2 or T_2 to include function symbols for polynomial hierarchy functions, we can still define σ. The defining equation for a function of the polynomial hierarchy must include an explicit bound on the size of the function. These bounds can be used to define σ.

Theorem 9 is stated only for S_2 and T_2; however our use of the σ metafunction means the proof holds for theories with a larger language.

Proof: of Theorem 9.

We shall give the proof for the theory S_2. Minor modifications are all that is needed to handle T_2 and we leave them to the reader.

By Proposition 8, there is a proof P of $\Gamma \longrightarrow \Delta$ with no free cuts and in free variable normal form. We shall modify P to be restricted by parameter variables.

Let the parameter variables of P be $c_1, \ldots, c_p$. Let $b_1, \ldots, b_n$ be the other free variables in P. Since P is in free variable normal form, each b_i has a unique elimination inference; we assume without loss of generality that if the elimination inference for b_i is below the elimination inference for b_j then $i<j$ (if not, reorder the b_i's). Note that two variables can not have the same elimination inference since we are assuming P is in free variable normal form.

We define $u_1, \ldots, u_n$ to be terms so that the free variables of u_i are the parameter variables $\vec{c}$. We define u_i by induction on i according to the following two cases.

(1) Suppose the elimination inference J_i of b_i is ($\forall\leq$:right) or ($\exists\leq$:left). That is, J_i is either

$$\frac{b_i \leq s_i(b_1, \ldots, b_{i-1}), \Gamma_i \longrightarrow A_i(b_i), \Delta_i}{\Gamma_i \longrightarrow (\forall x \leq s_i(b_1, \ldots, b_{i-1})) A_i(x), \Delta_i}$$

or

$$\frac{b_i \leq s_i(b_1, \ldots, b_{i-1}), A_i(b_i), \Gamma_i \longrightarrow \Delta_i}{(\exists x \leq s_i(b_1, \ldots, b_{i-1})) A_i(x), \Gamma_i \longrightarrow \Delta_i}$$

where the term s_i may contain the free variables $b_1, \ldots, b_{i-1}$ and may also contain the parameter variables $c_1, \ldots, c_p$. Then define $u_i = \sigma[s_i](u_1, \ldots, u_{i-1})$.

(2) Suppose the elimination inference J_i of b_i is an induction inference. So J_i is

$$\frac{\Gamma_i, A_i(\lfloor \frac{1}{2} b_i \rfloor) \longrightarrow A_i(b_i), \Delta_i}{\Gamma_i, A_i(0) \longrightarrow A_i(s_i(b_1, \ldots, b_{i-1})), \Delta_i}$$

Again, define $u_i = \sigma[s_i](u_1, \ldots, u_{i-1})$.

P will be modified to obtain a proof P^* with the same endsequent which is restricted by parameter variables. We will do this in two steps: first we form P' by changing each sequent in P; however, P' may not be a valid proof so we fix up the illegal inferences in P' to get P^*.

P' will have exactly the same structure as P and each sequent in P' is built from the corresponding sequent in P. Let $\Pi \longrightarrow \Lambda$ be a sequent in P. Let $b_{i_1}, \ldots, b_{i_m}$ be the free variables of P which have elimination inference below $\Pi \longrightarrow \Lambda$. Let Ξ be the cedent

$$b_{i_1} \le u_{i_1}, \ldots, b_{i_m} \le u_{i_m}.$$

The sequent in P' corresponding to $\Pi \longrightarrow \Lambda$ is $\Xi, \Pi \longrightarrow \Lambda$. So P' is formed from P by adding $b_i \le u_i$ to every sequent above the elimination inference of b_i, for $i=1, \ldots, m$. Thus the endsequent of P' is the same as the endsequent of P.

We now modify P' to obtain a valid proof P^*. It is easy to verify that there are exactly five ways in which P' fails to be a proof:

(1*) The initial sequents of P' are not valid initial sequents. An initial sequent of P' has the form

$$b_{i_1} \le u_{i_1}, \ldots, b_{i_m} \le u_{i_m}, \Pi \longrightarrow \Lambda$$

where $\Pi \longrightarrow \Lambda$ is a valid initial sequent. In P^*, this initial sequent is replaced by the initial sequent $\Pi \longrightarrow \Lambda$ and m (Weak:left) inferences.

(2*) Let I be a cut inference in P. The corresponding inference I' in P' will be of the form

$$\frac{\Xi, \Gamma \longrightarrow \Delta, A \qquad \Xi, A, \Pi \longrightarrow \Lambda}{\Xi, \Gamma, \Xi, \Pi \longrightarrow \Delta, \Lambda}$$

Unless Ξ is the empty cedent, this is not a valid inference. In P^* this inference is replaced by I^*:

$$\cfrac{\cfrac{\Xi,\Gamma \longrightarrow \Delta,A \qquad \Xi,A,\Pi \longrightarrow \Lambda}{\Xi,\Gamma,\Xi,\Pi \longrightarrow \Delta,\Lambda}}{\Xi,\Gamma,\Pi \longrightarrow \Delta,\Lambda}$$

where the double bar denotes a sequence of (Exchange:left) and (Contraction:left) inferences.

(3*) Let b_i be a free variable in P with a ($\forall\leq$:right) elimination inference J_i. The corresponding inference J_i' in P' is

$$\frac{b_i \leq u_i, \Xi_i, b_i \leq s_i, \Gamma_i \longrightarrow A_i(b_i), \Delta_i}{\Xi_i, \Gamma_i \longrightarrow (\forall x \leq s_i) A_i(x), \Delta_i}$$

where Ξ_i is the cedent containing the formulae $b_j \leq u_j$ for all b_j with elimination inference below J_i in P. Clearly, J_i' is not a valid inference. In P^* we replace J_i' by J_i^*:

$$\cfrac{\cfrac{\cfrac{\Xi_i, b_i \leq s_i \longrightarrow b_i \leq u_i \qquad b_i \leq u_i, \Xi_i, b_i \leq s_i, \Gamma_i \longrightarrow A_i(b_i), \Delta_i}{\Xi_i, b_i \leq s_i, \Xi_i, b_i \leq s_i, \Gamma_i \longrightarrow A_i(b_i), \Delta_i}}{\Xi_i, b_i \leq s_i, \Gamma_i \longrightarrow A_i(b_i), \Delta_i}}{\Xi_i, \Gamma_i \longrightarrow (\forall x \leq s_i) A_i(x), \Delta_i}$$

The first inference is a *Cut* inference. The sequent $\Xi_i, b_i \leq s_i \longrightarrow b_i \leq u_i$ is provable by the definition of the σ metafunction. The double bar between the second and third sequents indicates a sequence of inferences; in this case, a sequence of contraction and exchange inferences.

Since the cut inference is inessential it may be assumed without loss of generality to be free (since if not, it could be eliminated from the proof).

(4*) Suppose b_i is a free variable in P with a ($\exists\leq$:left) inference as its elimination inference J_i. We construct J_i^* as the corresponding inference in P^* by a construction similar to Case (3*).

(5*) Let b_i be a free variable in P with an induction inference J_i as its elimination inference. The corresponding inference J_i' in P' is:

$$\frac{b_i \leq u_i, \Xi_i, \Gamma_i, A_i(\lfloor \tfrac{1}{2} b_i \rfloor) \longrightarrow A_i(b_i), \Delta_i}{\Xi_i, \Gamma_i, A_i(0) \longrightarrow A_i(s_i), \Delta_i}$$

Clearly this is not a valid inference and in P^* we replace it by J_i^*:

$$
\begin{array}{lc}
 & b_i\le u_i \longrightarrow \lfloor \tfrac{1}{2}b_i\rfloor \le u_i \qquad b_i\le u_i,\Xi_i,\Gamma_i,A_i(\lfloor \tfrac{1}{2}b_i\rfloor)\longrightarrow A_i(b_i),\Delta_i \\
\hline
 & b_i\le u_i, b_i\le u_i,\Xi_i,\Gamma_i,\lfloor \tfrac{1}{2}b_i\rfloor\le u_i \supset A_i(\lfloor \tfrac{1}{2}b_i\rfloor)\longrightarrow A_i(b_i),\Delta_i \\
\hline
 & b_i\le u_i,\Xi_i,\Gamma_i,\lfloor \tfrac{1}{2}b_i\rfloor\le u_i \supset A_i(\lfloor \tfrac{1}{2}b_i\rfloor)\longrightarrow A_i(b_i),\Delta_i \\
\hline
 & \Xi_i,\Gamma_i,\lfloor \tfrac{1}{2}b_i\rfloor\le u_i \supset A_i(\lfloor \tfrac{1}{2}b_i\rfloor)\longrightarrow b_i\le u_i\supset A_i(b_i),\Delta_i \\
\hline
(\beta) & \Xi_i,\Gamma_i,\lfloor \tfrac{1}{2}b_i\rfloor\le\lfloor \tfrac{1}{2}d_i\rfloor,(\forall x\le\lfloor \tfrac{1}{2}d_i\rfloor)(x\le u_i\supset A_i(x))\longrightarrow b_i\le u_i\supset A_i(b_i),\Delta_i \\
\hline
 & \Xi_i,\Gamma_i,b_i\le d_i,(\forall x\le\lfloor \tfrac{1}{2}d_i\rfloor)(x\le u_i\supset A_i(x))\longrightarrow b_i\le u_i\supset A_i(b_i),\Delta_i \\
\hline
 & \Xi_i,\Gamma_i,(\forall x\le\lfloor \tfrac{1}{2}d_i\rfloor)(x\le u_i\supset A_i(x))\longrightarrow (\forall x\le d_i)(x\le u_i\supset A_i(x)),\Delta_i \\
\hline
(\gamma) & \Xi_i,\Gamma_i,(\forall x\le 0)(x\le u_i\supset A_i(x))\longrightarrow (\forall x\le u_i)(x\le u_i\supset A_i(x)),\Delta_i \\
\hline
(\delta) & \Xi_i,\Gamma_i,A_i(0)\longrightarrow (\forall x\le u_i)(x\le u_i\supset A_i(x)),\Delta_i \\
\hline
 & \Xi_i,\Xi_i,\Gamma_i,A_i(0)\longrightarrow A_i(s_i),\Delta_i \\
\hline\hline
 & \Xi_i,\Gamma_i,A_i(0)\longrightarrow A_i(s_i),\Delta_i
\end{array}
$$

where d_i is a new free variable and the sequents (β), (γ), and (δ) used in the *Cut* inferences are:

$$
\begin{aligned}
(\beta) &\quad = \quad b_i\le d_i\longrightarrow \lfloor \tfrac{1}{2}b_i\rfloor\le\lfloor \tfrac{1}{2}d_i\rfloor \\
(\gamma) &\quad = \quad A_i(0)\longrightarrow(\forall x\le 0)(x\le u_i\supset A_i(x)) \\
(\delta) &\quad = \quad \Xi_i,(\forall x\le u_i)(x\le u_i\supset A_i(x))\longrightarrow A_i(s_i)
\end{aligned}
$$

Note that these cuts are free since the cut formula is a direct descendent of an induction inference or of a formula appearing in an initial sequent. Also note that the *PIND* induction in P^* is restricted by parameter variables since the only free variables in u_i are $c_1,\ldots,c_p$.

This completes the construction of the desired proof P^*.

Q.E.D. □

It is not at all obvious that Theorem 9 holds for the theories S_2^i and T_2^i instead of S_2 and T_2. In fact, it almost certainly does not hold for S_2^0 and T_2^0. However, it does hold for S_2^i and T_2^i when $i>1$. The author surmises (without proof) that it holds for S_2^1 and T_2^1, but to prove this seems to require a more careful treatment of the foundations of Bounded Arithmetic than we gave in Chapter 2. At any rate, Theorem 9 as stated above suffices for our purposes in Chapter 7.

4.6. Polynomial Size, Induction Free Proofs.

This section establishes the following result: Suppose $A(\vec{a})$ is a bounded formula provable in S_2 where $\vec{a}$ indicates all the free variables of A. Then there is a deterministic polynomial time algorithm P such that for all $\vec{n} \in \mathbf{N}^p$, $P(\vec{n})$ is the Gödel number of a proof of $A(I_{n_1}, \ldots, I_{n_p})$, where the proof $P(\vec{n})$ is bounded and contains no induction inferences. To restate this informally, we can say that if A is bounded and if $S_2 \vdash (\forall \vec{x}) A(\vec{x})$ then for each n there is a "short," bounded, induction free proof of $A(\vec{n})$.

The results of this section are interesting in their own right; however, we wish to apply them in Chapter 7 to Gödel incompleteness theorems. Accordingly, it is important to note that all the proof theoretic arguments below are constructive and part of these arguments can be formalized in S_2^1.

Theorem 10: Let $\Gamma \longrightarrow \Delta$ be a bounded sequent provable in S_2. Let $a_1, \ldots, a_p$ be the free variables in $\Gamma \longrightarrow \Delta$. Then there is a p-ary polynomial time function f such that for all $\vec{n} \in \mathbf{N}^p$, $f(\vec{n})$ is the Gödel number of an S_2-proof of $\Gamma(I_{n_1}, \ldots, I_{n_p}) \longrightarrow \Delta(I_{n_1}, \ldots, I_{n_p})$ which is bounded, does not contain any induction inferences and is in free variable normal form.

Recall that I_n is a term with value n such that the length of I_n is proportional to $|n|$. The theorem would certainly be false if $S^{(n)}0$ were used instead of I_n since the length of $S^{(n)}0$ is exponential in the length of n.

Proof: By Theorem 9 there is a bounded proof P of $\Gamma \longrightarrow \Delta$ which is restricted by parameter variables and is in free variable normal form. The idea behind the theorem is that given values $n_1, \ldots, n_p$ for $a_1, \ldots, a_p$, we can expand each induction inference in P into a series of cuts.

The proof of Theorem 10 is by induction on the number of inferences of P. The only interesting case to consider is when the final inference of P is an induction inference; so let the final inference in P have the form

$$\frac{\Gamma, A(\lfloor \frac{1}{2} b \rfloor) \longrightarrow A(b), \Delta}{\Gamma, A(0) \longrightarrow A(t(\vec{a})), \Delta}$$

where the only free variables in t are the $\vec{a}$. We eliminate the induction inference by replacing it with $2 \cdot |t(\vec{n})| \dot{-} 1$ *Cut* inferences. Specifically, if m is the value of $t(\vec{n})$, form the $|m|+1$ terms $I_0, \ldots, I_{MSP(m,i)}, \ldots, I_m$. By the induction hypothesis, there is a deterministic polynomial time function $h(\vec{a}, b)$ which computes the Gödel number for an induction free, bounded proof in free variable normal form of $\Gamma, A(\lfloor \frac{1}{2} I_b \rfloor) \longrightarrow A(I_b), \Delta$. By invoking h repeatedly we can obtain proofs of each of the sequents

$$\Gamma, A(\lfloor \tfrac{1}{2} I_{MSP(m,i)} \rfloor) \longrightarrow A(I_{MSP(m,i)}), \Delta.$$

It is also easy to construct a proof of $A(I_{MSP(m,i+1)}) \longrightarrow A(\lfloor \frac{1}{2} I_{MSP(m,i)} \rfloor)$ for all i. Then we join these sequents together with $2\cdot|m|\dot{-}1$ cuts (and a lot of exchanges and contractions) to obtain a proof of $\Gamma,A(0)\longrightarrow A(I_m),\Delta$. Since for every term t there is a polynomial p_t such that $p_t(|\vec{n}|)\geq|t(\vec{n})|$ for all $\vec{n}$, this procedure is a polynomial time procedure.

It is also important to see that if t is any term, then there is a deterministic polynomial time function g_t such that $g_t(\vec{n})$ is the Gödel number of an induction free proof of $I_m=t(I_{n_1},\ldots,I_{n_p})$. We shall prove this last sentence as part of Lemma 7.5. Thus there is a polynomial time procedure which produces an induction free proof of

$$A(I_m)\longrightarrow A(t(I_{n_1},\ldots,I_{n_p})).$$

We combine this with the proof of $\Gamma,A(0)\longrightarrow A(I_m),\Delta$ obtained above. This yields an induction free, bounded proof of $\Gamma,A(0)\longrightarrow A(t(I_{n_1},\ldots,I_{n_p})),\Delta$. By renaming free variables we can ensure that the proof is in free variable normal form.

Q.E.D. □

4.7. Parikh's Theorem.

The next theorem is originally due to Parikh [20]. Parikh gave a proof-theoretic proof and, later, a simpler model-theoretic proof was found. However, we present a proof theoretic proof here since we have already developed most of the necessary machinery anyway.

If a theory proves $(\forall x)(\exists y)A(x,y)$ we regard this as a proof that there is a total function f such that for all x, $A(x,f(x))$ holds. Parikh's theorem states that a function defined in this way can be bounded by a term of Bounded Arithmetic, provided that A is a bounded formula.

Theorem 11: (Parikh) Let $i\geq0$. Suppose that A is a bounded formula and that S_2^i or T_2^i proves $(\forall\vec{x})(\exists y)A(\vec{x},y)$. Then there is a term $r(\vec{x})$ such that the same theory proves $(\forall\vec{x})(\exists y\leq r(\vec{x}))A(\vec{x},y)$.

Proof: By Proposition 8 there must be a free cut free proof P in free variable normal form of the sequent $\longrightarrow(\exists y)A(\vec{c},y)$. It is easily seen that every formula in P is either $(\exists y)A(\vec{c},y)$ or is bounded. Furthermore, every occurrence of $(\exists y)A(\vec{c},y)$ is in the antecedent. Thus the only inferences in P involving unbounded quantifiers are ($\exists$:right) inferences which introduce the formula $(\exists y)A(\vec{c},y)$.

We modify the proof P as follows:

Step (1): First, we will mimic the proof of Theorem 9 to obtain a proof P''. Let all notation be as in the proof of Theorem 9. The construction of P' can be carried out on P since the only unbounded quantifier inferences of P are ($\exists$:right) inferences and since P is in free variable normal form. P'' is obtained from P' in much the same way as P^* is. Recall that

P^* was defined by the Cases (1^*)–(5^*). P'' is defined from P' by Cases $(1'')$–$(5'')$. Cases $(1'')$–$(4'')$ are the same as (1^*)–(4^*). The fifth case is:

$(5'')$ Suppose the inference J_i' in P' is:

$$\frac{b_i \le u_i, \Xi_i, \Gamma_i, A_i(\lfloor \frac{1}{2} b_i \rfloor) \longrightarrow A_i(b_i), \Delta_i}{\Xi_i, \Gamma_i, A_i(0) \longrightarrow A_i(s_i), \Delta_i}$$

Clearly this is not a valid inference and in P'' we replace it by J_i'':

$$\dfrac{\dfrac{(\delta) \quad \dfrac{(\gamma) \quad \dfrac{\dfrac{\dfrac{b_i \le u_i \longrightarrow \lfloor \frac{1}{2} b_i \rfloor \le u_i \qquad b_i \le u_i, \Xi_i, \Gamma_i, A_i(\lfloor \frac{1}{2} b_i \rfloor) \longrightarrow A_i(b_i), \Delta_i}{b_i \le u_i, b_i \le u_i, \Xi_i, \Gamma_i, \lfloor \frac{1}{2} b_i \rfloor \le u_i \supset A_i(\lfloor \frac{1}{2} b_i \rfloor) \longrightarrow A_i(b_i), \Delta_i}}{b_i \le u_i, \Xi_i, \Gamma_i, \lfloor \frac{1}{2} b_i \rfloor \le u_i \supset A_i(\lfloor \frac{1}{2} b_i \rfloor) \longrightarrow A_i(b_i), \Delta_i}}{\Xi_i, \Gamma_i, \lfloor \frac{1}{2} b_i \rfloor \le u_i \supset A_i(\lfloor \frac{1}{2} b_i \rfloor) \longrightarrow b_i \le u_i \supset A_i(b_i), \Delta_i}}{\Xi_i, \Gamma_i, \lfloor \frac{1}{2} 0 \rfloor \le u_i \supset A_i(\lfloor \frac{1}{2} 0 \rfloor) \longrightarrow s_i \le u_i \supset A_i(s_i), \Delta_i}}{\Xi_i, \Gamma_i, A_i(0) \longrightarrow s_i \le u_i \supset A_i(s_i), \Delta_i}}{\Xi_i, \Xi_i, \Gamma_i, A_i(0) \longrightarrow A_i(s_i), \Delta_i}$$
$$\overline{\overline{\Xi_i, \Gamma_i, A_i(0) \longrightarrow A_i(s_i), \Delta_i}}$$

where (γ) is the sequent $A_i(0) \longrightarrow \lfloor \frac{1}{2} 0 \rfloor \le u_i \supset A_i(\lfloor \frac{1}{2} 0 \rfloor)$ and (δ) represents

$$\frac{\Xi_i \longrightarrow s_i \le u_i \qquad A_i(s_i) \longrightarrow A_i(s_i)}{\Xi_i, s_i \le u_i \supset A_i(s_i) \longrightarrow A_i(s_i)}$$

It is easy to verify that P'' is free cut free and in free variable normal form and that the endsequent of P'' is the same as the endsequent of P.

Step (2): Wherever a ($\exists$:right) inference occurs in P'', of the form

$$\frac{\Xi, \Gamma \longrightarrow \Delta, A(\vec{c}, t(\vec{b}))}{\Xi, \Gamma \longrightarrow \Delta, (\exists y) A(\vec{c}, y)}$$

We replace this inference with:

$$\dfrac{\Xi \longrightarrow t(\vec{b}) \le \sigma[t](\vec{u}) \qquad \dfrac{\Xi, \Gamma \longrightarrow \Delta, A(\vec{c}, t(\vec{b}))}{t(\vec{b}) \le \sigma[t](\vec{u}), \Xi, \Gamma \longrightarrow \Delta, (\exists y \le \sigma[t](\vec{u})) A(\vec{c}, y)}}{\Xi, \Xi, \Gamma \longrightarrow \Delta, (\exists y \le \sigma[t](\vec{u})) A(\vec{c}, y)}$$
$$\overline{\overline{\Xi, \Gamma \longrightarrow \Delta, (\exists y \le \sigma[t](\vec{u})) A(\vec{c}, y)}}$$

We also replace all the descendants of $(\exists y)A(\vec{c},y)$ in P'' by $(\exists y \le \sigma[t](\vec{u}))A(\vec{c},y)$ as far down as possible: which means all the descendants either down to the end of P'' or down to a contraction inference with $(\exists y)A(\vec{c},y)$ as principal formula.

Step (3): After doing Step (2) as often as possible, we handle contractions. Suppose the modified proof contains

$$\frac{\Xi,\Gamma \longrightarrow \Delta,(\exists y \le t)A(\vec{c},y),(\exists y \le s)A(\vec{c},y)}{\Xi,\Gamma \longrightarrow \Delta,(\exists y)A(\vec{c},y)}$$

We replace this by first deriving

$$(\exists y \le s)A(\vec{c},y) \longrightarrow (\exists y \le \sigma[s,t])A(\vec{c},y)$$

and

$$(\exists y \le t)A(\vec{c},y) \longrightarrow (\exists y \le \sigma[s,t])A(\vec{c},y)$$

We now use two cuts and a contraction to get:

$$\cfrac{\cfrac{\cfrac{\Xi,\Gamma \longrightarrow \Delta,(\exists y \le t)A(\vec{c},y),(\exists y \le s)A(\vec{c},y)}{\Xi,\Gamma \longrightarrow \Delta,(\exists y \le \sigma[s,t])A(\vec{c},y),(\exists y \le s)A(\vec{c},y)}}{\Xi,\Gamma \longrightarrow \Delta,(\exists y \le \sigma[s,t])A(\vec{c},y),(\exists y \le \sigma[s,t])A(\vec{c},y)}}{\Xi,\Gamma \longrightarrow \Delta,(\exists y \le \sigma[s,t])A(\vec{c},y)}$$

We now replace the descendants of the original formula $(\exists y)A(\vec{c},y)$ as far down as possible in the proof, just as we did in Step (2).

We iterate Step (3) as often as possible.

The end result of the above construction is a proof of $(\exists y \le r)A(\vec{c},y)$ for some term r.

Q.E.D. □

The restriction in Parikh's theorem that A be a bounded formula is necessary as the following counterexample shows. Let A be the formula

$$A(x,y) \;=\; (\forall z)(|z|=x \supset |y|=x)$$

Then *LKB* proves $(\forall x)(\exists y)A(x,y)$. But there is no term r of Bounded Arithmetic such that $(\forall x)(\exists y \le r)A(x,y)$ is true.

Chapter 5

Computational Complexity of Definable Functions

This chapter is concerned with establishing the converse to Theorem 3.1, which stated that any function in $\square_i^p = PTC(\Sigma_{i-1}^p)$ can be Σ_i^b–defined in S_2^i. Theorem 3.1 was proved by a straightforward construction of the Σ_i^b–formula from the definition of a $\square_i^p$–function. The converse is a deeper result and its proof depends strongly on the cut-elimination theorem.

This chapter deals only with first order theories of arithmetic. Second order theories of arithmetic are treated in Chapters 9 and 10.

Theorem 1: (The Main Theorem). Let $i \geq 1$. Suppose $S_2^i \vdash (\forall \vec{x})(\exists y)A(\vec{x},y)$ where $A(\vec{x},y)$ is a Σ_i^b–formula and $\vec{x}$ and y are the only free variables of A. Then there is a term $t(\vec{x})$, a Σ_i^b–formula B and a function g in $\square_i^p$ so that

(1) $S_2^i \vdash (\forall \vec{x})(\forall y)(B(\vec{x},y) \supset A(\vec{x},y))$
(2) $S_2^i \vdash (\forall \vec{x})(\forall y)(\forall z)(B(\vec{x},y) \wedge B(\vec{x},z) \supset y=z)$
(3) $S_2^i \vdash (\forall \vec{x})(\exists y \leq t)B(\vec{x},y)$
(4) For all $\vec{n}$, $\mathbf{N} \models B(\vec{n},g(\vec{n}))$.

Corollary 2: Suppose f is a function Σ_i^b–definable by S_2^i. Then f is a $\square_i^p$–function.

Corollary 2 is an immediate consequence of Theorem 1. The proof of Theorem 1 is the rest of this chapter.

5.1. Witnessing a Bounded Formula.

Before we can prove Theorem 1, we need some preliminary definitions.

Definition: Suppose $i \geq 1$ and A is a Σ_i^b–formula and $\vec{a}$ is a vector of free variables which includes all the variables free in A. We define below a formula $Witness_A^{i,\vec{a}}(w,\vec{a})$ which is Δ_i^b with respect to S_2^i. The definition is by induction on the complexity of A.

(1) If A is a Σ_{i-1}^b– or a Π_{i-1}^b–formula, then we define

$$Witness_A^{i,\vec{a}}(w,\vec{a}) \iff A(\vec{a})$$

(2) If A is $B \wedge C$, then we define

$$Witness_A^{i,\vec{a}}(w,\vec{a}) \iff Witness_B^{i,\vec{a}}(\beta(1,w),\vec{a}) \wedge Witness_C^{i,\vec{a}}(\beta(2,w),\vec{a})$$

(3) If A is $B \vee C$, then we define

$$Witness_A^{i,\vec{a}}(w,\vec{a}) \iff Witness_B^{i,\vec{a}}(\beta(1,w),\vec{a}) \vee Witness_C^{i,\vec{a}}(\beta(2,w),\vec{a})$$

(4) If A is not in $\Sigma_{i-1}^b \cup \Pi_{i-1}^b$ and $A(\vec{a})$ is $(\forall x \le |s(\vec{a})|)B(\vec{a},x)$, then we define

$$\begin{aligned} Witness_A^{i,\vec{a}}(w,\vec{a}) \iff & Seq(w) \wedge Len(w) = |s(\vec{a})|+1 \wedge \\ & \wedge (\forall x \le |s(\vec{a})|)\, Witness_{B(\vec{a},b)}^{i,\vec{a},b}(\beta(x+1,w),\vec{a},x) \end{aligned}$$

Thus w witnesses $A(\vec{a})$ iff $w = \langle w_0, \ldots, w_{|s|} \rangle$ and each w_i witnesses $B(\vec{a},i)$.

(5) If A is not in $\Sigma_{i-1}^b \cup \Pi_{i-1}^b$ and A is $(\exists x \le t(\vec{a}))B(\vec{a},x)$, then we define

$$\begin{aligned} Witness_A^{i,\vec{a}}(w,\vec{a}) \iff & Seq(w) \wedge Len(w) = 2 \wedge \beta(1,w) \le t(\vec{a}) \wedge \\ & \wedge Witness_{B(\vec{a},b)}^{i,\vec{a},b}(\beta(2,w),\vec{a},\beta(1,w)) \end{aligned}$$

So w witnesses A iff $w = \langle n,v \rangle$ where $n \le t$ and v witnesses $B(\vec{a},n)$.

(6) If A is not in $\Sigma_{i-1}^b \cup \Pi_{i-1}^b$ and A is $\neg B$, then we define $Witness_A^{i,\vec{a}}$ by using logical prenex operations to transform A so that it can be handled by cases (1)-(5). Specifically, if A is $\neg(\neg B)$, $\neg(B \wedge C)$, $\neg(B \vee C)$, $\neg(\forall x \le t)B$ or $\neg(\exists x \le t)B$ then let A^* be B, $(\neg B) \vee (\neg C)$, $(\neg B) \wedge (\neg C)$, $(\exists x \le t)(\neg B)$ or $(\forall x \le t)(\neg B)$ respectively. Then

$$Witness_A^{i,\vec{a}}(w,\vec{a}) \iff Witness_{A^*}^{i,\vec{a}}(w,\vec{a})$$

The idea behind defining $Witness_A^{i,\vec{a}}$ is that having a w such that $Witness_A^{i,\vec{a}}(w,\vec{a})$ is a canonical way of verifying that $A(\vec{a})$ is true. It is not difficult to see that $(\exists w) Witness_A^{i,\vec{a}}(w,\vec{a})$ is equivalent to $A(\vec{a})$ when $A \in \Sigma_i^b$. Indeed, this is provable by S_2^i:

Proposition 3: Let $i \ge 1$. Let A be any Σ_i^b-formula with free variables among $\vec{a}$. Then:

(a) $S_2^1 \vdash Witness_A^{i,\vec{a}}(w,\vec{a}) \supset A(\vec{a})$

(b) There is a term t_A such that

$$S_2^i \vdash A(\vec{a}) \supset (\exists w \le t_A(\vec{a}))\, Witness_A^{i,\vec{a}}(w,a).$$

(c) Furthermore, there is a $\square_1^p$–function g_A which is Σ_1^b–definable in S_2^1 such that

$$S_2^1 \vdash Witness_A^{i,\vec{a}}(w,\vec{a}) \supset Witness_A^{i,\vec{a}}(g_A(w),\vec{a}) \wedge g_A(w) \le t_A(\vec{a}).$$

Proof:

(a) This is easy to show by induction on the complexity of A.

(b) This is also proved by induction on the complexity of A. Cases (1)-(3) and (6) of the definition of $Witness_A^{i,\vec{a}}$ are easily handled. The other two cases are as follows:

Case (5): $A \notin \Sigma_{i-1}^b \cup \Pi_{i-1}^b$ and A is $(\exists x \le t)B(\vec{a},x)$. We argue informally in S_2^i. Suppose $B(\vec{a},x)$ holds with $x \le t$. By the induction hypothesis, we know that there exists a v such that $Witness_{B(\vec{a},b)}^{i,\vec{a},b}(v,\vec{a},x)$. So let $w = \langle x,v \rangle$. Then $Witness_A^{i,\vec{a}}(w,\vec{a})$ holds. We can define

$$t_A(\vec{a}) = SqBd(\max(t_B(\vec{a},t(\vec{a})),t(\vec{a})),2)$$

and we are guaranteed that $w \le t_A(\vec{a})$.

Case (4): $A \notin \Sigma_{i-1}^b \cup \Pi_{i-1}^b$ and A is $(\forall x \le |s(\vec{a})|)B(\vec{a},x)$. The induction hypothesis is that

$$S_2^i \vdash B(\vec{a},b) \supset (\exists w \le t_B(\vec{a},b))\, Witness_B^{i,\vec{a},b}(w,\vec{a},b).$$

Since the Σ_i^b–replacement axioms are theorems of S_2^i (by Theorem 2.14), it follows that

$$S_2^i \vdash A(\vec{a}) \supset (\exists w \le SqBd(\sigma[t_B](\vec{a},|s|),s))\, Witness_A^{i,\vec{a}}(w,\vec{a}).$$

(c) This is easily proved by induction on the complexity of A. The essential idea is that sequences can be coded efficiently.

Q.E.D. □

Another crucial property of *Witness* is that it is relatively easy to tell whether $Witness_A^{i,\vec{c}}(w,\vec{c})$ holds for arbitrary w and $\vec{c}$. This is formalized by the next proposition.

Proposition 4: Let $i \ge 1$ and $A(\vec{c}) \in \Sigma_i^b$. Let p be the predicate defined by

$$p(w,\vec{c}) \iff Witness_A^{i,\vec{c}}(w,\vec{c})$$

Then p is a Δ_i^p–predicate (of the polynomial hierarchy).

Proof: This is easily proved by induction on the complexity of A. □

In particular, when $i=1$ $p(w,\vec{c})$ is a polynomial time predicate. This should not be surprising since if A is a fixed Σ_1^b–formula it is certainly reasonable that a polynomial time algorithm can check whether w and $\vec{c}$ code an instantiation for A which satisfies A. Of course, this polynomial time algorithm depends on A.

If Γ is a cedent we write $\bigwedge\Gamma$ (respectively, $\bigvee\Gamma$) to denote the conjunction (respectively, disjunction) of the formulae in Γ. We adopt the convention that conjunction and disjunction associate from right to left. Thus, if Γ is A,B,C then $\bigwedge\Gamma$ means $A\wedge(B\wedge C)$. We use the notation

$$\ll a_1, \ldots, a_n \gg$$

to denote $<a_1,<a_2, \ldots ,<a_{n-1},a_n> \cdots >>$.

These conventions allow us to conveniently discuss witnessing a cedent. For example, suppose Γ is $A_1, \ldots ,A_n$ and that $w=\ll w_1, \ldots ,w_n\gg$. Then $Witness_{\bigwedge\Gamma}^{i,\vec{a}}(w,\vec{a})$ holds iff $Witness_{A_j}^{i,\vec{a}}(w_j,\vec{a})$ holds for each positive $j\leq n$.

5.2. The Main Proof.

We shall prove Theorem 1 by proving a more general theorem:

Theorem 5: Let $i\geq 1$. Suppose $S_2^i\vdash\Gamma,\Pi\longrightarrow\Lambda,\Delta$ and that each formula in $\Gamma\cup\Delta$ is a Σ_i^b–formula and each formula in $\Pi\cup\Lambda$ is a Π_i^b–formula. Let $c_1, \ldots ,c_p$ be the free variables in $\Gamma,\Pi\longrightarrow\Lambda,\Delta$. Let G and H be the Σ_i^b–formulae

$$G = (\bigwedge\Gamma)\wedge\bigwedge\{\neg C : C\in\Lambda\}$$

and

$$H = (\bigvee\Delta)\vee\bigvee\{\neg C : C\in\Pi\}.$$

Then there is a function f which is Σ_i^b–definable in S_2^i such that

(1) f is a $\square_i^p$–function, and

(2) $S_2^i\vdash Witness_G^{i,\vec{c}}(w,\vec{c})\supset Witness_H^{i,\vec{c}}(f(w,\vec{c}),\vec{c})$.

Proof: of Theorem 1 from 5:

The hypothesis of Theorem 1 is that $S_2^i \vdash (\exists y)A(\vec{c},y)$. Hence, by Theorem 4.11 there is a term $t(\vec{c})$ such that $S_2^i \vdash (\exists y \le t)A(\vec{c},y)$. We now apply Theorem 5 by letting Δ be $(\exists y \le t)A(\vec{c},y)$ and letting $\Gamma = \Pi = \Lambda = \emptyset$. Theorem 5 asserts that there is an f satisfying (1) and (2). Furthermore this f is Σ_i^b-definable in S_2^i by $f(\vec{c}) = d \iff A_f(\vec{c},d)$ for some $A \in \Sigma_i^b$ such that

$$S_2^i \vdash (\forall \vec{x})(\exists ! y)A_f(x,y).$$

We need to find B and g satisfying (1)-(4) of Theorem 1. We define

$$g(\vec{a}) \;=\; \beta(1,f(\vec{a}))$$

and

$$B(\vec{x},y) \iff y = \beta(1,f(\vec{x})).$$

It now follows immediately from the definition of *Witness* and Proposition 3 that g and B satisfy the conclusions of Theorem 1. Note that g is a $\square_i^p$-function since f is.

Q.E.D. □

Proof: of Theorem 5.

By Proposition 4.8, there is an S_2^i-proof P of $\Gamma,\Pi \longrightarrow \Lambda,\Delta$ such that P is free cut free and in free variable normal form. In particular, since every formula in the endsequent of P is in $\Sigma_i^b \cup \Pi_i^b$, so is every formula appearing anywhere in P. Since all induction inferences in P are Σ_i^b-*PIND* inferences, the principal formula of each cut inference in P is a Σ_i^b-formula.

To simplify notation and terminology we shall henceforth assume that Π and Λ are the empty cedent. We can always fulfill this requirement by using ($\neg$:left) and ($\neg$:right) inferences to move formulae from side to side. Furthermore, no essential cases are ignored under this assumption since each inference has a dual; for example, ($\exists\le$:left) is dual to ($\forall\le$:right) and ($\wedge$:right) is dual to ($\vee$:left).

The proof of Theorem 5 is by induction on the number of inferences in the proof P of $\Gamma \longrightarrow \Delta$ where P is assumed to be free cut free and in free variable normal form.

To begin, consider the case where P has no inferences and consists of a single sequent. Then $\Gamma \longrightarrow \Delta$ must be either a *BASIC* axiom, a logical axiom or an equality axiom. In either case every formula in $\Gamma \longrightarrow \Delta$ is open. The definition of *Witness* was that

$$Witness_A^{i,\vec{a}} \iff A(\vec{a})$$

whenever A is open. Thus, conditions (1) and (2) of Theorem 5 are satisfied if we choose f to be the constant zero function.

The argument for the induction step splits into thirteen cases depending on what the last inference of P is:

Case (1): Suppose the last inference of P is ($\neg$:left) or ($\neg$:right). These are "cosmetic" inferences: see the discussion above about assuming that Π and Λ are empty.

Case (2): ($\wedge$:left). Suppose the last inference of P is

$$\frac{B,\Gamma^* \longrightarrow \Delta}{B \wedge C,\Gamma^* \longrightarrow \Delta}$$

Let D be the formula $B\wedge(\bigwedge\Gamma^*)$ and let E be $(B\wedge C)\wedge(\bigwedge\Gamma^*)$. The induction hypothesis is that there is a $\square_i^p$–function g such that g is Σ_i^b–definable by S_2^i and

$$S_2^i \vdash \mathit{Witness}_D^{i,\vec{c}}(w,\vec{c}) \supset \mathit{Witness}_{\bigvee\Delta}^{i,\vec{c}}(g(w,\vec{c}),\vec{c}).$$

Let h be the function defined by

$$h(w) \;=\; <\beta(1,\beta(1,w)),\beta(2,w)>.$$

Then $h\in\square_1^p$ and

$$S_2^i \vdash \mathit{Witness}_E^{i,\vec{c}}(w,\vec{c}) \supset \mathit{Witness}_D^{i,\vec{c}}(h(w),\vec{c})$$

follows immediately from the definition of *Witness*. So define $f(w,\vec{c})$ to be $g(h(w),\vec{c})$. Then $f\in\square_i^p$, f is Σ_i^b–definable and

$$S_2^i \vdash \mathit{Witness}_E^{i,\vec{c}}(w,\vec{c}) \supset \mathit{Witness}_{\bigvee\Delta}^{i,\vec{c}}(f(w,\vec{c}),\vec{c})$$

which is what we needed to show.

Case (3): ($\vee$:left). Suppose the last inference of P is

$$\frac{B,\Gamma^* \longrightarrow \Delta \qquad\qquad C,\Gamma^* \longrightarrow \Delta}{B \vee C,\Gamma^* \longrightarrow \Delta}$$

Let D be the formula $B\wedge(\bigwedge\Gamma^*)$, let E be $C\wedge(\bigwedge\Gamma^*)$ and let F be $(B\vee C)\wedge(\bigwedge\Gamma^*)$. By the induction hypothesis, there are Σ_i^b–definable functions g and h in $\square_i^p$ such that

$$S_2^i \vdash Witness_D^{i,\vec{c}}(w,\vec{c}) \supset Witness_{\bigvee\Delta}^{i,\vec{c}}(g(w,\vec{c}),\vec{c})$$

and

$$S_2^i \vdash Witness_E^{i,\vec{c}}(w,\vec{c}) \supset Witness_{\bigvee\Delta}^{i,\vec{c}}(h(w,\vec{c}),\vec{c}).$$

We define f as

$$f(w,\vec{c}) = \begin{cases} g(<\beta(1,\beta(1,w)),\beta(2,w)>,\vec{c}) & \text{if } Witness_B^{i,\vec{c}}(\beta(1,\beta(1,w)),\vec{c}) \\ h(<\beta(2,\beta(1,w)),\beta(2,w)>,\vec{c}) & \text{otherwise} \end{cases}$$

The idea is that if w witnesses $(B \vee C) \wedge (\bigwedge\Gamma^*)$ then either $\beta(1,\beta(1,w))$ witnesses B or $\beta(2,\beta(1,w))$ witnesses C. In the former case, g is used to find a witness for $\bigvee\Delta$; in the latter case, h is used. This can easily be formalized in S_2^i, so

$$S_2^i \vdash Witness_F^{i,\vec{c}}(w,\vec{c}) \supset Witness_{\bigvee\Delta}^{i,\vec{c}}(f(w,\vec{c}),\vec{c}).$$

Now f is a $\square_i^p$–function since g and h are and by Proposition 4. Also, f is Σ_i^b–definable by S_2^i since g and h are and since $Witness_B^{i,\vec{c}}$ is a Δ_i^b–predicate.

Case (4): ($\exists\leq$:left). Suppose the last inference of P is

$$\frac{a \leq t, B(a), \Gamma^* \longrightarrow \Delta}{(\exists x \leq t)B(x), \Gamma^* \longrightarrow \Delta}$$

Of course, a is an eigenvariable and must not appear in the lower sequent. Let D be the formula $a \leq t \wedge (B(a) \wedge (\bigwedge\Gamma^*))$, and let E be $(\exists x \leq t)B(x) \wedge (\bigwedge\Gamma^*)$. By the induction hypothesis, there is a Σ_i^b–definable function $g \in \square_i^p$ such that

$$S_2^i \vdash Witness_D^{i,\vec{c},a}(w,\vec{c},a) \supset Witness_{\bigvee\Delta}^{i,\vec{c}}(g(w,\vec{c},a),\vec{c}).$$

(Note that we can omit the variable a from the superscript on the righthand side of the implication since a does not appear free in Δ.)

First consider the case where $(\exists x \leq t)B$ is not in Σ_{i-1}^b. Define the function h by

$$h(w) \;=\; \ll 0,\beta(2,\beta(1,w)),\beta(2,w) \gg .$$

By the definition of *Witness* we have

$$S_2^i \vdash Witness_E^{i,\vec{c}}(w,\vec{c}) \supset Witness_D^{i,\vec{c},a}(h(w),\vec{c},\beta(1,\beta(1,w))).$$

So define f by

$$f(w,\vec{c}) \;=\; g(h(w),\vec{c},\beta(1,\beta(1,w))).$$

Thus f is a $\square_i^p$-function, f is Σ_i^b-definable by S_2^i and

$$S_2^i \vdash Witness_E^{i,\vec{c}}(w,\vec{c}) \supset Witness_{\bigvee\Delta}^{i,\vec{c}}(f(w,\vec{c}),\vec{c}).$$

The case where $(\exists x \le t)B \in \Sigma_{i-1}^b$ is even easier. We now let

$$h(w) \;=\; \ll 0,0,\beta(2,w) \gg$$

and

$$f(w,\vec{c}) \;=\; g(h(w),\vec{c},\,(\mu x \le t)B(x)\,).$$

Note $f \in \square_i^p$ since $(\mu x \le t)B(x)$ can be computed either by using a binary search or, when $(\exists x \le t)$ is a sharply bounded quantifier, by an exhaustive search.

Case (5): ($\forall\le$:left). Suppose the last inference of P is

$$\frac{B(s),\Gamma^* \longrightarrow \Delta}{s \le t,(\forall x \le t)B(x),\Gamma^* \longrightarrow \Delta}$$

We shall assume that $s \le t$ is in Γ (a similar argument works for $s \le t$ in Π.) Let D be the formula $B(s) \wedge (\bigwedge\Gamma^*)$, and let E be $s \le t \wedge (\forall x \le t)B(x) \wedge (\bigwedge\Gamma^*)$. The induction hypothesis is that there is a Σ_i^b-definable function g in $\square_i^p$ such that

$$S_2^i \vdash Witness_D^{i,\vec{c}}(w,\vec{c}) \supset Witness_{\bigvee\Delta}^{i,\vec{c}}(g(w,\vec{c}),\vec{c}).$$

First consider the case where $(\forall x \le t)B(x)$ is not in $\Sigma_{i-1}^b \cup \Pi_{i-1}^b$. Then $(\forall x \le t)$ must be a sharply bounded quantifier with $t = |r|$ for some term r. Define the function h by

$$h(w,\vec{c}) = <\beta(s(\vec{c})+1,\beta(1,\beta(2,w))),\beta(2,\beta(2,w))>.$$

By the definition of *Witness*, we have

$$S_2^i \vdash Witness_E^{i,\vec{c}}(w,\vec{c}) \supset Witness_D^{i,\vec{c}}(h(w,\vec{c}),\vec{c}).$$

So define $f(w,\vec{c}) = g(h(w,\vec{c}),\vec{c})$. It is straightforward to see that f satisfies the desired conditions of Theorem 5.

The case where $(\forall x \le t)B(x) \in \Sigma_{i-1}^b \cup \Pi_{i-1}^b$ is easier. We now set $h(w,\vec{c})$ equal to $<0,\beta(2,\beta(2,w))>$ and otherwise proceed as before.

Case (6): ($\supset$:left) and ($\supset$:right). We omit these cases: see ($\vee$:left) and ($\vee$:right).

Case (7): ($\vee$:right). Suppose the last inference of P is

$$\frac{\Gamma \longrightarrow B, \Delta^*}{\Gamma \longrightarrow B \vee C, \Delta^*}$$

Let D be the formula $B \vee (\bigvee \Delta^*)$. By the induction hypothesis, there is a Σ_i^b–definable function g in $\square_i^p$ such that

$$S_2^i \vdash Witness_{\bigwedge\Gamma}^{i,\vec{c}}(w,\vec{c}) \supset Witness_D^{i,\vec{c}}(g(w,\vec{c}),\vec{c}).$$

Define h by

$$h(w) \;=\; <<\beta(1,w),0>,\beta(2,w)>$$

and let $f(w,\vec{c}) = h(g(w,c))$. Then it is easy to see that f satisfies all the desired conditions.

Case (8): ($\wedge$:right). Suppose the last inference of P is

$$\frac{\Gamma \longrightarrow B, \Delta^* \qquad\qquad \Gamma \longrightarrow C, \Delta^*}{\Gamma \longrightarrow B \wedge C, \Delta^*}$$

Let D be the formula $B \vee (\bigvee \Delta^*)$, let E be $C \vee (\bigvee \Delta^*)$ and let F be $(B \wedge C) \vee (\bigvee \Delta^*)$. The induction hypothesis is that there are $\square_i^p$–functions g and h which are Σ_i^b–definable by S_2^i such that

$$S_2^i \vdash Witness_{\bigwedge\Gamma}^{i,\vec{c}}(w,\vec{c}) \supset Witness_D^{i,\vec{c}}(g(w,\vec{c}),\vec{c})$$

and

$$S_2^i \vdash Witness_{\bigwedge\Gamma}^{i,\vec{c}}(w,\vec{c}) \supset Witness_E^{i,\vec{c}}(h(w,\vec{c}),\vec{c}).$$

We define the function k as

$$k(v,w,\vec{c}) \;=\; \begin{cases} v & \text{if } Witness_{\bigvee\Delta^*}^{i,\vec{c}}(v,\vec{c}) \\ w & \text{otherwise} \end{cases}$$

By Proposition 4, k is a $\square_i^p$–function; also, k is Σ_i^b–definable by S_2^i since $Witness_{\bigvee\Delta^*}^{i,\vec{c}}$ is a Δ_i^b–formula. Now define f by

$$f(w,\vec{c}) \;=\; <<\beta(1,g(w,\vec{c})),\beta(1,h(w,\vec{c}))>,k(\beta(2,g(w,\vec{c})),\beta(2,h(w,\vec{c})),\vec{c})>.$$

Clearly f is Σ_i^b–definable by S_2^i and is in $\square_i^p$, since g, h, and k are. Also, it is easy to see that

$$S_2^i \vdash Witness_{\bigwedge\Gamma}^{i,\vec{c}}(w,\vec{c}) \supset Witness_F^{i,\vec{c}}(f(w,\vec{c}),\vec{c}).$$

Case (9): ($\exists\le$:right). Suppose the last inference of P is

$$\frac{\Gamma^* \longrightarrow B(s),\Delta^*}{s\le t,\Gamma^* \longrightarrow (\exists x\le t)B(x),\Delta^*}$$

We shall assume that $s\le t$ is in Γ (a similar argument works for $s\le t$ in Π). Let D be the formula $B(s)\vee(\bigvee\Delta^*)$, let E be $s\le t\wedge(\bigwedge\Gamma^*)$ and let F be $(\exists x\le t)B(x)\vee(\bigvee\Delta^*)$. The induction hypothesis is that there is a $\square_i^p$–function g which is Σ_i^b–definable in S_2^i such that

$$S_2^i \vdash Witness_{\bigwedge\Gamma^*}^{i,\vec{c}}(w,\vec{c}) \supset Witness_D^{i,\vec{c}}(g(w,\vec{c}),\vec{c}).$$

By the definition of *Witness*,

$$S_2^i \vdash Witness_E^{i,\vec{c}}(w,\vec{c}) \supset s\le t \wedge Witness_{\bigwedge\Gamma^*}^{i,\vec{c}}(\beta(2,w),\vec{c}).$$

So define f to be

$$f(w,\vec{c}) \;=\; <<s(\vec{c}),\beta(1,g(\beta(2,w),\vec{c}))>,\beta(2,g(\beta(2,w),\vec{c}))>.$$

Then f is Σ_i^b–definable by S_2^i, f is a $\square_i^p$–function and

$$S_2^i \vdash Witness_E^{i,\vec{c}}(w,\vec{c}) \supset Witness_F^{i,\vec{c}}(f(w,\vec{c}),\vec{c}).$$

Case (10): ($\forall\le$:right). Suppose the last inference of P is

$$\frac{a\le t,\Gamma \longrightarrow B(a),\Delta^*}{\Gamma \longrightarrow (\forall x\le t)B(x),\Delta^*}$$

where a is the eigenvariable and does not appear free in the lower sequent. Let D be the formula $a \le t \wedge (\bigwedge\Gamma)$, let E be $B(a) \vee (\bigvee\Delta^*)$ and let $F(\vec{c},d)$ be $(\forall x \le d)B(x) \vee (\bigvee\Delta^*)$. The induction hypothesis is that there is a $\square_i^p$–function g such that

$$S_2^i \vdash Witness_D^{i,\vec{c},a}(w,\vec{c},a) \supset Witness_E^{i,\vec{c},a}(g(w,\vec{c},a),\vec{c},a).$$

First, consider the case where $(\forall x \le t)B(x)$ is not in $\Sigma_{i-1}^b \cup \Pi_{i-1}^b$. So $(\forall x \le t)$ is sharply bounded with $t = |r|$ for some term r. We define the function k by

$$k(v,w,\vec{c}) = \begin{cases} v & \text{if } Witness_{\bigvee\Delta^*}^{i,\vec{c}}(v,\vec{c}) \\ w & \text{otherwise} \end{cases}$$

We define f by the following limited iteration scheme:

$$\begin{aligned} p(w,\vec{c},0) &= <<\beta(1,g(w,\vec{c},0))>,\beta(2,g(w,\vec{c},0))> \\ p(w,\vec{c},m+1) &= <\beta(1,p(w,\vec{c},m))*\beta(1,g(w,\vec{c},m+1)), \\ &\qquad k(\beta(2,p(w,\vec{c},m)),\beta(2,g(w,\vec{c},m+1)),\vec{c})> \\ f(w,\vec{c}) &= p(<0,w>,\vec{c},|r|) \end{aligned}$$

By Proposition 4, $k \in \square_i^p$ and hence $f \in \square_i^p$. It is straightforward to see that

$$S_2^i \vdash Witness_D^{i,\vec{c}}(w,\vec{c}) \supset Witness_F^{i,\vec{c},d}(p(w,\vec{c},0),\vec{c},0)$$

and

$$\begin{aligned} S_2^i \vdash Witness_D^{i,\vec{c}}(w,\vec{c}) \wedge Witness_F^{i,\vec{c},d}(p(w,\vec{c},d),\vec{c},d) \supset \\ \supset Witness_F^{i,\vec{c},d}(p(w,\vec{c},d+1),\vec{c},d+1). \end{aligned}$$

It follows by Σ_i^b–$LIND$ that

$$S_2^i \vdash Witness_{\bigwedge\Gamma}^{i,\vec{c}}(w,\vec{c}) \supset Witness_F^{i,\vec{c},d}(f(w,\vec{c}),\vec{c},t).$$

Hence,

$$S_2^i \vdash Witness_{\bigwedge\Gamma}^{i,\vec{c}}(w,\vec{c}) \supset Witness_{F(\vec{c},t)}^{i,\vec{c}}(f(w,\vec{c}),\vec{c})$$

which is what we needed to show.

Second, consider the case where $(\forall x \le t)B(x)$ is in $\Sigma_{i-1}^b \cup \Pi_{i-1}^b$. If $A(\vec{c},a)$ is any one of the formulae $a \le t$, $B(a)$ or $(\forall x \le t)B(x)$ then $Witness_A^{i,\vec{c},a}(w,\vec{c},a)$ is defined to be equivalent to $A(\vec{c},a)$ itself. Let $h(w,\vec{c})$ be the $\square_i^p$–function $(\mu x \le t)B(x)$ and let

$f(w,\vec{c})=g(<0,w>,\vec{c},h(w,\vec{c}))$. Then f satisfies the desired conditions.

Case (11): (*Cut*). Suppose the last inference of P is

$$\frac{\Gamma \longrightarrow B,\Delta \qquad\qquad B,\Gamma \longrightarrow \Delta}{\Gamma \longrightarrow \Delta}$$

Since P is free cut free, B must be a Σ_i^b–formula. Let D be the formula $B\lor(\bigvee\Delta)$ and let E be $B\land(\bigwedge\Gamma)$. The induction hypothesis is that there are $\square_i^p$–functions g and h which are Σ_i^b–defined by S_2^i such that

$$S_2^i \vdash Witness_{\bigwedge\Gamma}^{i,\vec{c}}(w,\vec{c}) \supset Witness_D^{i,\vec{c}}(g(w,\vec{c}),\vec{c})$$

and

$$S_2^i \vdash Witness_E^{i,\vec{c}}(w,\vec{c}) \supset Witness_{\bigvee\Delta}^{i,\vec{c}}(h(w,\vec{c}),\vec{c}).$$

We define the function f as

$$f(w,\vec{c}) \;=\; \begin{cases} \beta(2,g(w,\vec{c})) & \text{if } \; Witness_{\bigvee\Delta}^{i,\vec{c}}(\beta(2,g(w,\vec{c})),\vec{c}) \\ h(<\beta(1,g(w,\vec{c})),w>,\vec{c}) & \text{otherwise} \end{cases}$$

By Proposition 4, $f\in\square_i^p$, and since $Witness_{\bigvee\Delta}^{i,\vec{c}}$ is Δ_i^b with respect to S_2^i, f is Σ_i^b–defined by S_2^i. Also, it is easy to see that

$$S_2^i \vdash Witness_{\bigwedge\Gamma}^{i,\vec{c}}(w,\vec{c}) \supset Witness_{\bigvee\Delta}^{i,\vec{c}}(f(w,\vec{c}),\vec{c}).$$

Case (12): (Σ_i^b–*PIND*). Suppose the last inference of P is

$$\frac{B(\lfloor \tfrac{1}{2}a \rfloor),\Gamma^* \longrightarrow B(a),\Delta^*}{B(0),\Gamma^* \longrightarrow B(t),\Delta^*}$$

where a is an eigenvariable and must not appear in the lower sequent. We shall only consider the case where $B(0)$ is in Γ and $B(t)$ is in Δ. (If this is not the case, then $B\in\Sigma_{i-1}^b\cup\Pi_{i-1}^b$ and the argument is much simpler.)

The general idea of the argument for Case (12) is to treat the Σ_i^b–*PIND* inference as if it were $|t|\dot{-}1$ cuts. So, in effect, Case (12) is handled by iterating the method of Case (11).

Let D be the formula $B(\lfloor\tfrac{1}{2}a\rfloor)\land(\bigwedge\Gamma^*)$, let E be $B(a)\lor(\bigvee\Delta^*)$, let F be $B(0)\land(\bigwedge\Gamma^*)$ and let $A(\vec{c},d)$ be $B(d)\lor(\bigvee\Delta^*)$. The induction hypothesis is that there is a $\square_i^p$–function g

such that

$$S_2^i \vdash Witness_D^{i,\vec{c},a}(w,\vec{c},a) \supset Witness_E^{i,\vec{c},a}(g(w,\vec{c},a),\vec{c},a).$$

We define $\square_i^p$–functions k and h by

$$k(v,w,\vec{c}) = \begin{cases} v & \text{if } Witness_{\bigvee\Delta^*}^{i,\vec{c}}(v,\vec{c}) \\ w & \text{otherwise} \end{cases}$$

$$h(v,w,\vec{c},a) = g(<\beta(1,v),\beta(2,w)>,\vec{c},a).$$

By Proposition 3(c) there is a term t_A and a $\square_1^p$–function q which is Σ_1^b–definable in S_2^1 such that

$$S_2^i \vdash Witness_A^{i,\vec{c},d}(w,\vec{c},d) \supset Witness_A^{i,\vec{c},d}(q(w),\vec{c},d) \wedge q(w) \leq t_A(\vec{c},d).$$

Now define f^* by the following limited iteration scheme:

$$p(w,\vec{c},0) = q(<\beta(1,w),0>)$$

$$\begin{aligned} p(w,\vec{c},m+1) = q(&<\beta(1,h(p(w,\vec{c},m),w,\vec{c},MSP(t,|t|\dot{-}(m+1)))), \\ &k(\beta(2,p(w,\vec{c},m)),\beta(2,h(p(w,\vec{c},m),w,\vec{c},MSP(t,|t|\dot{-}(m+1)))),\vec{c})>) \end{aligned}$$

$$f^*(w,\vec{c},u) = p(w,\vec{c},|u|).$$

This is a valid limited iteration definition since the use of the function q gives a provable polynomial space bound on p; namely, $p(w,\vec{c},m) \leq \sigma[t_A](\vec{c},t)$. Thus f^* is a $\square_i^p$–function which is Σ_i^b–definable in S_2^i.

Now it is easy to see that

$$S_2^i \vdash Witness_F^{i,\vec{c}}(w,\vec{c}) \supset Witness_A^{i,\vec{c},d}(f^*(w,\vec{c},0),\vec{c},0)$$

and

$$\begin{aligned} S_2^i \vdash Witness_F^{i,\vec{c}}(w,\vec{c}) \wedge Witness_A^{i,\vec{c},d}(f^*(w,\vec{c},\lfloor\tfrac{1}{2}u\rfloor),\vec{c},MSP(t,|t|\dot{-}|\lfloor\tfrac{1}{2}u\rfloor|)) \supset \\ \supset Witness_A^{i,\vec{c},d}(f^*(w,\vec{c},u,),\vec{c},MSP(t,|t|\dot{-}|u|)). \end{aligned}$$

So by Σ_i^b–$PIND$ with respect to u,

$$S_2^i \vdash Witness_F^{i,\vec{c}}(w,\vec{c}) \supset Witness_A^{i,\vec{c},d}(f^*(w,\vec{c},t),\vec{c},t).$$

So define $f(w,\vec{c})=f^*(w,\vec{c},t)$ and we are done.

Case (13): (Structural inference). The cases where the last inference of P is a weak inference, an exchange inference or a contraction inference are all trivial and we omit their proofs.

Q.E.D. □

5.3. The Main Theorem for First Order Bounded Arithmetic.

Combining Theorem 1 with Theorem 3.1 we get:

Theorem 6: Let $i \geq 1$. Suppose A is a Σ_i^b–formula and that $S_2^i \vdash (\forall \vec{x})(\exists y)A(\vec{x},y)$. Then there is a term t, a Σ_i^b–formula B and a function $f \in \square_i^p$ such that

(1) $S_2^i \vdash (\forall \vec{x})(\exists y \leq t)B(\vec{x},y)$
(2) $S_2^i \vdash (\forall \vec{x})(\forall y)(B(\vec{x},y) \supset A(\vec{x},y))$
(3) $S_2^i \vdash (\forall \vec{x})(\forall y)(\forall z)(B(\vec{x},y) \wedge B(\vec{x},z) \supset y=z)$
(4) For all $\vec{n}$, $\mathbf{N} \models B(\vec{n},f(\vec{n}))$

Conversely, if $f \in \square_i^p$, then there is a term t and a Σ_i^b–formula B such that (1), (3) and (4) hold.

Corollary 7: Let $i \geq 1$. A function f is Σ_i^b–definable in S_2^i iff $f \in \square_i^p$.

For the special case $i=1$, we have

Corollary 8: The Σ_1^b–definable functions of S_2^1 are precisely the polynomial time computable functions.

We can restate Theorem 6 in terms of predicates instead of functions as follows:

Theorem 9: Let $i \geq 1$. Suppose A is a Σ_i^b–formula, B is a Π_i^b–formula and that $S_2^i \vdash A(\vec{a}) \leftrightarrow B(\vec{a})$. Then there is a predicate $Q \in \Delta_i^p$ such that for all $\vec{n}$,

$$Q(\vec{n}) \iff \mathbf{N} \models A(\vec{n}) \iff \mathbf{N} \models B(\vec{n}).$$

Conversely, if $Q \in \Delta_i^p$, then there are formulae A and B so that all of the above holds.

Proof: This is an immediate consequence of Theorem 6. It is proved by noting that the function

$$f(\vec{x}) = \begin{cases} 0 & \text{if } A(\vec{x}) \\ 1 & \text{otherwise} \end{cases}$$

is Σ_1^b–definable in S_2^1 by the equation

$$f(\vec{x})=y \iff [y=0 \wedge A(\vec{x})] \vee [y=1 \wedge \neg B(\vec{x})].$$

Thus $f \in \square_1^p$ and hence A represents a predicate in P. □

Recall that in Chapter 1 we characterized the NP predicates as those expressible by Σ_1^b–formulae and the co–NP predicates as those expressible by Π_1^b–formulae. Hence in the case $i=1$, Theorem 9 becomes:

Corollary 10: Let $A(\vec{a})$ be a formula such that S_2^1 proves A is equivalent to a Σ_1^b– and to a Π_1^b–formula (i.e., S_2^1 proves that $A \in NP \cap$co–NP). Then $A(\vec{a})$ is a polynomial time predicate (i.e., A is in P).

So any predicate which is S_2^1–provably in $NP \cap$co–NP is in P.

5.4. Relativization.

The results proved above can be relativized by introducing oracles. For this two things must be done: firstly, enlarge the language of Bounded Arithmetic to include new function symbols for oracles, and secondly, use oracle Turing machines for computations.

We relativize the theories S_2^i in the following way:

Definition: Let $k \geq 1$ and let $p(n_1, \ldots, n_k)$ be a suitable polynomial. For each $j \geq 0$, $\eta_{j,k}^p$ is a k-ary function symbol. The *bounding axiom* for $\eta_{j,k}^p$ is

$$|\eta_{j,k}^p(a_1, \ldots, a_k)| \leq p(|a_1|, \ldots, |a_k|).$$

Definition: Let $\eta_{j_1,k_1}^{p_1}, \ldots, \eta_{j_n,k_n}^{p_n}$ be a sequence of function symbols. We write $\vec{\eta}$ as an abbreviation for that sequence. The theory $S_2^i(\vec{\eta})$ is defined to be the theory with the language of Bounded Arithmetic plus the symbols $\eta_{j_1,k_1}^{p_1}, \ldots, \eta_{j_n,k_n}^{p_n}$ and with the following axioms:

(1) the *BASIC* axioms

(2) for each $1 \le t \le n$, the bounding axiom: $|\eta_{j_t,k_t}^{p_t}(\vec{a})| \le p_t(|\vec{a}|)$

(3) the $\Sigma_i^b(\vec{\eta})$–*PIND* axioms.

Recall that in Chapter 1 ω_k^p was defined to be the set of k-ary functions with growth rate bounded by the polynomial p.

Definition: If $\eta_{j,k}^p$ is a function symbol then the *function space associated with* $\eta_{j,k}^p$ is ω_k^p.

We can relativize Theorem 1 as follows:

Theorem 11: Let $\vec{\eta}$ be a vector of function symbols and let $\vec{\omega}=\omega_{k_1}^{p_1}, \ldots ,\omega_{k_n}^{p_n}$ be the vector of function spaces associated with the η's. Let $i \ge 1$ be fixed.

Suppose $S_2^i(\vec{\eta}) \vdash (\forall \vec{x})(\exists y)A(\vec{x},y)$ where A is a $\Sigma_i^b(\vec{\eta})$–formula and $\vec{x}$ and y are the only free variables in $A(\vec{x},y)$. Then there is a term $t(\vec{x})$, a $\Sigma_i^b(\vec{\eta})$–formula B and a functional g in $\square_i^p(\vec{\omega})$ so that

(1) $S_2^i(\vec{\eta}) \vdash (\forall \vec{x})(\forall y)(B(\vec{x},y) \supset A(\vec{x},y))$

(2) $S_2^i(\vec{\eta}) \vdash (\forall \vec{x})(\forall y)(\forall y)(B(\vec{x},y) \wedge B(\vec{x},z) \supset y=z)$

(3) $S_2^i(\vec{\eta}) \vdash (\forall \vec{x})(\exists y \le t(\vec{x}))B(\vec{x},y)$

(4) For all $\vec{n} \in \mathbf{N}^n$ and all oracles $\Omega_1, \ldots ,\Omega_n$ with $\Omega_i \in \omega_{k_i}^{p_i}$ for all $1 \le i \le n$,

$$(\mathbf{N},\vec{\Omega}) \models B(\vec{n},g(\vec{n},\vec{\Omega})).$$

Proof: The entire proof of Theorem 1 including Theorem 5 can be relativized. This yields a proof of Theorem 11. □

Corollary 12: Suppose A and B are Δ_1^b–formulae with respect to S_2^1, q is a suitable polynomial and that

$$S_2^1 \vdash (\forall x)(\exists y < 2^{q(|x|)})B(x,y) \supset (\forall x)(\exists y)A(x,y).$$

Then there is a functional $g \in \square_1^p(\omega_1^q)$ such that whenever $\Omega \in \omega_1^q$,

$$(\mathbf{N},\Omega) \models (\forall x)B(x,\Omega(x)) \supset (\forall x)A(x,g(x,\Omega)).$$

Recall that the condition that $g \in \square_1^p(\omega_1^q)$ means that g can be computed by a deterministic, polynomial-time, oracle Turing machine M_g where the function oracle Ω used by M_g is required to satisfy $|\Omega(x)| \leq q(|x|)$ for all x.

Proof: By the hypothesis of the corollary,

$$S_2^1(\eta_{1,1}^q) \vdash (\forall x) B(x, \eta_{1,1}^q(x)) \supset (\forall x)(\exists y) A(x,y).$$

So,

$$S_2^1(\eta_{1,1}^q) \vdash (\forall x)(\exists y)\,[A(x,y) \vee \neg B(y, \eta_{1,1}^q(y))].$$

Then, by Theorem 11 there is a $g \in \square_1^p(\omega_1^q)$ such that, for all x and all $\Omega \in \omega_1^q$, $g(x,\Omega)$ is equal to either a y such that $A(x,y)$ holds or a y such that $B(y,\Omega(y))$ fails.

Q.E.D. □

Definition: Let f be a unary function symbol. Then $PHP(f)$ is an abbreviation for the formula

$$a \neq 0 \wedge (\forall y < 2a)(f(y) < a) \supset (\exists y < 2a)(\exists z < 2a)(f(y) = f(z) \wedge y \neq z).$$

So $PHP(f)$ expresses a pigeon hole principle stating that $2 \cdot a$ pigeons can not sit in a holes. Note that a appears as a free variable in $PHP(f)$.

Corollary 13: $S_2^1(f) \nvdash PHP(f)$.

Of course, $S_2^1(f)$ means the theory extending S_2^1 with the new function symbol f and the $\Sigma_1^b(f)$*-PIND* axioms.

Proof: Suppose the corollary is false, then let q be the polynomial $q(n) = n$. Then

$$S_2^1(\eta_{1,1}^q) \vdash PHP(\eta_{1,1}^q).$$

Hence,

$$S_2^1(\eta_{1,1}^q) \vdash a \neq 0 \supset (\exists y < 2a)(\exists z < 2a)\,[\eta_{1,1}^q(y) \geq a \vee (\eta_{1,1}^q(y) = \eta_{1,1}^q(z) \wedge y \neq z)].$$

So by Theorem 11 there is an $f \in \square_1^p(\omega_1^q)$ such that for all $a \in \mathbf{N}$ and all oracles $\Omega \in \omega_1^q$, $f(a,\Omega) = \langle y,z \rangle$ where y and z satisfy the above condition.

But this is absurd. f is computed by a polynomial time, oracle Turing machine M_f, so $M_f(x,\Omega)$ has run time $\leq p(|x|)$ for all x and some polynomial p. Choose x_0 large enough so that $x_0 > p(|x_0|) + 2$. Then define Ω_0 so that the following conditions hold:

If $M_f(x_0,\Omega_0)$ first queries its oracle for the value of $\Omega_0(m)$ on the n-th step where $m<2\cdot x_0$, then set $\Omega_0(m)$ to be equal to the greatest number $j\leq\min(m,x_0-3)$ such that no earlier oracle query of $M_f(x_0,\Omega_0)$ yielded the answer j. Such a j will always exist.

If $M_f(x_0,\Omega_0)=<y_0,z_0>$ and if $\Omega_0(y_0)$ and/or $\Omega_0(z_0)$ have not yet been defined, set $\Omega_0(y_0)=x_0-1$ and/or set $\Omega_0(z_0)=x_0-2$.

For all other values of m, set $\Omega_0(m)=0$.

Q.E.D. □

Corollary 13 states that $S_2^1(f)$ can not prove the pigeon hole principle *PHP(f)*. On the other hand, Alex Wilkie [30] showed that $S_2(f)$ can prove *PHP(f)*. Examining Wilkie's proof closely yields the following theorem:

Theorem 14: (Wilkie [30]). $T_2^2(f)\vdash PHP(f)$.

Combining Wilkie's theorem and Corollary 13 gives

Corollary 15: $T_2^2(f)$ is not equivalent to $S_2^1(f)$.

It is an open question whether S_2^1 is equivalent to T_2^2 or even if S_2^1 is equivalent to S_2.

Chapter 6

Cook's Equational Theory PV

PV is an equational theory of polynomial time functions introduced by Cook [6]. *PV* contains a schema which allows function symbols to be introduced for each polynomial time function and an induction schema which is essentially equivalent to the *PIND* axioms applied to open formulae of *PV*.

Our earlier results have shown that S_2^1 can Σ_1^b-define precisely the polynomial time functions. Thus it is not too surprising that S_2^1 and *PV* are closely related. We shall see below that, after making allowances for the fact that they have different languages, S_2^1 and *PV* have the same Σ_1^b-formulae as theorems.

6.1. Preliminaries for PV and PV1.

Like S_2^1, the universe of *PV* is the nonnegative integers. *PV* codes integers by dyadic coding, as used by Smullyan [25]. An integer n is represented by the string $d_k d_{k-1} \cdots d_0$ where $n=\sum_{i=0}^{k} 2^i \cdot d_i$ and each d_i is either 1 or 2.

PV has two unary functions s_1 and s_2 which are helpful for handling dyadic notation. They are defined by

$$s_i(d_k d_{k-1} \cdots d_0) = d_k d_{k-1} \cdots d_0 i$$

i.e., $s_i(x)=2x+i$.

PV has other initial function symbols in addition to s_1 and s_2, see [6] for details. *PV* can also introduce new function symbols by a schema which Cook calls *limited recursion on notation*, but in the terminology of this dissertation is more appropriately called *limited iteration on notation*. Suppose g, h_1, h_2, k_1 and k_2 have already been introduced as *PV*-function symbols. Then we can define a new *PV*-function symbol f by

$$\begin{aligned} f(0,\vec{y}) &= g(\vec{y}) \\ f(s_i(x),\vec{y}) &= h_i(x,\vec{y},f(x,\vec{y})) \end{aligned}$$

provided that *PV* proves

$$|h_i(x,\vec{y},z)|_d \leq |z|_d + |k_i(x,\vec{y})|_d$$

for i=1,2. Here we are introducing $|x|_d$ as a function whose value is equal to the length of the dyadic code for x, namely $|x|_d = \lfloor \log_2(x+1) \rfloor$. The fact that this inequality is expressible in PV is proved by Cook [6].

It is clear that limited iteration on notation as defined above is similar to the limited iteration defined in Chapter 1. Hence, by Cobham [5], a function symbol for each polynomial time function can be introduced in PV.

PV has only one predicate symbol, namely = (equality). However, we shall follow the convention that a function symbol can be interpreted as a predicate by letting a nonzero value denote *True* and a zero value denote *False*. For example, Cook [6] defines the function $PROOF$ so that

$$PROOF(m,n) = \begin{cases} 1 & \text{if } m \text{ is the Gödel number of an equation and} \\ & \quad n \text{ is the Gödel number of a } PV\text{–proof of } m. \\ 0 & \text{otherwise} \end{cases}$$

In [6] it is asserted that many function symbols can be introduced in PV. In addition to the function symbols defined there, PV has symbols for the functions S, $+$, $\cdot$, $\#$, $|x|$ and $\lfloor \frac{1}{2}x \rfloor$ as well as functions for handling sequences; namely, $\beta(i,w)$, the pairing function

$$(w_1,w_2) \mapsto\ <w_1,w_2>$$

and the sequence extension function $*$

$$<a_1,\ldots,a_n>*a_{n+1} \ = \ <a_1,\ldots,a_{n+1}>.$$

(Our definition for $*$ conflicts with the notation in [6]. Our function $*$ is completely distinct from Cook's.) Furthermore, PV can prove all the simple properties of these functions; in particular, PV can prove all the *BASIC* axioms.

The syntax of PV can be expanded to allow quantifier free logical formulae instead of just equations. Cook [6] gives a detailed description of how this may be done and he calls the enlarged theory $PV1$. We shall not distinguish between PV and $PV1$ notationally and we shall continue to refer to the enlarged system as PV.

We also enlarge the syntax of PV to allow the predicate symbol $\leq$. Of course this is just an extension by definitions: $x \leq y$ denotes the formula $LE(x,y) \neq 0$ where LE is the PV–function symbol defined in [6] so that $LE(x,y)=1$ if $x \leq y$ and otherwise $LE(x,y)=0$.

In addition to the binary length function $|x|$, PV can define the dyadic length function $|x|_d$ by limited iteration on notation:

$$|0|_d = 0$$

$$|s_1(x)|_d = |s_2(x)|_d = |x|_d+1$$

PV can prove the simple properties of length functions including the formulae $|x| \geq |x|_d$, $|x| \leq |x|_d+1$, and $|x+1| = |x|_d+1$.

PV defines function symbols corresponding to the logical operators:

$$NOT(x) = \begin{cases} 0 & \text{if } x \neq 0 \\ 1 & \text{if } x = 0 \end{cases}$$

$$AND(x,y) = \begin{cases} 0 & \text{if } x=0 \text{ or } y=0 \\ 1 & \text{otherwise} \end{cases}$$

$$OR(x,y) = NOT(AND(NOT(x),NOT(y)))$$

We can, in effect, use sharply bounded quantifiers in PV by introducing new function symbols which have a similar effect:

Definition: Let $P(z,\vec{x})$ and $F(\vec{x})$ be PV-function symbols. Then

$$Q(\vec{x}) = (\overline{\forall} z \leq |F(\vec{x})|)P(z,\vec{x})$$

is a PV-function symbol so that

$$Q(\vec{x}) = \begin{cases} 1 & \text{if } (\forall z \leq |F(\vec{x})|)(0 \neq P(z,\vec{x})) \\ 0 & \text{otherwise} \end{cases}$$

$Q(\vec{x})$ is defined by the following limited iteration on notation scheme:

$$f(0,\vec{x},z) = NOT(NOT(P(0,\vec{x})))$$

$$f(s_i(y),\vec{x},z) = AND(f(y,\vec{x},z), OR(P(|s_1(y)|_d,\vec{x}),LE(z,|y|_d)))$$

$$Q(\vec{x}) = f(s_1(F(\vec{x})),\vec{x},|F(\vec{x})|)$$

Also, $(\overline{\exists} z \leq |F(\vec{x})|)P(z,\vec{x})$ is defined to be the PV-function symbol $NOT((\overline{\forall} z \leq |F(\vec{x})|)NOT(P(z,\vec{x})))$.

Proposition 1: Let $G(\vec{x},y)$ be any PV–function symbol. Then there is a PV–function symbol $F(\vec{x},y)$ so that

$$PV \vdash \beta(0,F(\vec{x},y))=|y|+1 \wedge 0 \neq (\overline{\forall} z \leq |y|)(G(\vec{x},z)=\beta(z+1,F(\vec{x},y))).$$

Proposition 1 states that PV satisfies an analogue of the Δ_1^b–replacement property, in that the value of $F(\vec{x},y)$ is $<G(\vec{x},0),\ldots,G(\vec{x},|y|)>$.

Proof: Let $H(\vec{x},y,z)$ be the PV–function symbol defined by the following limited iteration on notation:

$$H(\vec{x},y,0) = <G(\vec{x},0)>$$

$$H(\vec{x},y,s_i(w)) = \begin{cases} H(\vec{x},y,w)*G(\vec{x},|w|_d+1) & \text{if } |w|_d<|y| \\ H(\vec{x},y,w) & \text{otherwise} \end{cases}$$

Now set $F(\vec{x},y)=H(\vec{x},y,s_1(y))$. □

6.2. $\mathbf{S_2^1}$ and the Language of PV.

In order to state the conservation results concerning S_2^1 and PV, we must enlarge the language of S_2^1 to include the language of PV. First we note:

Proposition 2: S_2^1 can Σ_1^b–define all the functions of PV.

Proof: This is proved just like Theorem 3.1. Indeed, there is no substantive difference between limited iteration and limited iteration on notation. □

Definition: L_{PV} is the set of non-logical symbols of PV. Let $S_2^1(L_{PV})$ be the theory containing S_2^1 and the language L_{PV}. In addition, for each function symbol F in L_{PV}, $S_2^1(L_{PV})$ has a Σ_1^b–defining axiom for F which defines F in terms of its limited iteration definition.

In other words, $S_2^1(L_{PV})$ is S_2^1 plus symbols for the Σ_1^b–defined functions of PV. Proposition 2 guarantees that $S_2^1(L_{PV})$ can be so defined.

It is immediately obvious that $S_2^1(L_{PV})$ is a stronger theory than PV. This is because the axioms of PV are all theorems of $S_2^1(L_{PV})$. In particular, the induction on notation axioms of PV are Δ_1^b–$PIND$ axioms of $S_2^1(L_{PV})$.

We shall need the following axiomatization of $S_2^1(L_{PV})$:

Definition: $S_2^1(PV)$ is the theory with the same language as $S_2^1(L_{PV})$ and with the following axioms:

(1) The open *BASIC* axioms of S_2^1,

(2) The $\Sigma_1^b(L_{PV})$-*PIND* axioms,

(3) The following axioms defining the initial function symbols of PV (compare with [6]):

$$\begin{aligned}
&s_1(x)=2x+1\\
&s_2(x)=2x+2\\
&TR(0)=0\\
&TR(s_i(x))=x\\
&x \circledast 0=x\\
&x \circledast s_i(y)=s_i(x \circledast y)\\
&x \otimes 0=0\\
&x \otimes s_i(y)=x \circledast (x \otimes y)\\
&LESS(x,0)=x\\
&LESS(x,s_i(y))=TR(LESS(x,y))
\end{aligned}$$

where i=1,2 and where we are using $\circledast$ to denote Cook's function $*$ since $*$ has already been used for other purposes.

(4) Whenever f is a defined function symbol of PV, introduced by equations (2.2)-(2.4) of Cook [6], $S_2^1(PV)$ includes the axioms

$$\begin{aligned}
&f(0,\vec{y})=g(\vec{y})\\
&f(s_i(x),\vec{y})=h_i(x,\vec{y},f(x,\vec{y}))
\end{aligned}$$

Proposition 3: $S_2^1(L_{PV})$ and $S_2^1(PV)$ are equivalent theories.

Proof: It is clear that $S_2^1(L_{PV})$ is stronger than $S_2^1(PV)$. For the converse, it is necessary to show that $S_2^1(PV)$ proves that every $f \in L_{PV}$ satisfies the Σ_1^b-defining axiom $f(\vec{x})=y \Longleftrightarrow A_f(\vec{x},y)$ by which f is defined in $S_2^1(L_{PV})$. This is easily shown as follows: we can introduce a new function f' in $S_2^1(PV)$ by defining f' to satisfy the Σ_1^b-defining axiom $f'(\vec{x})=y \Longleftrightarrow A_f(\vec{x},y)$. Then this theory $S_2^1(PV,f')$ is a conservative extension of $S_2^1(PV)$. It is now easy to prove by *PIND* that $(\forall \vec{x})(f(\vec{x})=f'(\vec{x}))$, and hence f satisfies the defining equation for f'. □

We next state the main theorem of this chapter. (This theorem was independently conjectured by Stephen Cook.)

Theorem 4: Let $t=u$ be any equation of PV. Then $S_2^1(PV)\vdash t=u$ iff $PV\vdash t=u$.

One direction of this theorem is immediate from our remark above that $S_2^1(L_{PV})$ is stronger than PV. To prove the converse we shall show below that the results of Chapter 5 can be partially formalized in PV.

6.3. Witnessing a Σ_1^b-Formula.

For the sake of avoiding excessive subscripts, we use $\Sigma_i^b(PV)$ and $\Pi_i^b(PV)$ as synonyms for $\Sigma_i^b(L_{PV})$ and $\Pi_i^b(L_{PV})$ from now on.

In order to handle $\Sigma_1^b(PV)$–formulae in the theory PV, we need a way for PV to assert that a given $\Sigma_1^b(PV)$–formula is true.

Definition: Let A be a $\Sigma_1^b(PV)$–formula and $\vec{a}$ be a vector of k free variables containing all the free variables of A. $WITNESS_A^{\vec{a}}$ is a $(k+1)$-ary function symbol of PV defined by induction on the complexity of A as follows:

(1) If A is atomic,

$$WITNESS_A^{\vec{a}}(w,\vec{a}) = \begin{cases} 1 & \text{if } A(\vec{a}) \\ 0 & \text{otherwise} \end{cases}$$

(2) If A is $B\wedge C$,

$$WITNESS_A^{\vec{a}}(w,\vec{a}) = AND(WITNESS_B^{\vec{a}}(\beta(1,w),\vec{a}), WITNESS_C^{\vec{a}}(\beta(2,w),\vec{a}))$$

(3) If A is $B\vee C$,

$$WITNESS_A^{\vec{a}}(w,\vec{a}) = OR(WITNESS_B^{\vec{a}}(\beta(1,w),\vec{a}), WITNESS_C^{\vec{a}}(\beta(2,w),\vec{a}))$$

(4) If A is $(\forall x\leq|t|)B(x,\vec{a})$ where $B(b,\vec{a})\in\Sigma_1^b(PV)$, then

$$WITNESS_A^{\vec{a}}(w,\vec{a}) = (\overline{\forall} x\leq|t|)\, WITNESS_B^{b,\vec{a}}(\beta(x+1,w),x,\vec{a})$$

(5) If A is $(\exists x\leq t)B(x,\vec{a})$ where $B\in\Sigma_1^b(PV)$, then

$$WITNESS_A^{\vec{a}}(w,\vec{a}) = WITNESS_B^{b,\vec{a}}(\beta(2,w),\beta(1,w),\vec{a})\wedge\beta(1,w)\leq t$$

(6) If A is $\neg B$ then we transform A by logical operations so that Cases (1)-(5) apply. Specifically, if A is $\neg(\neg B)$, $\neg(B\vee C)$, $\neg(B\wedge C)$, $\neg(\forall x\leq t)B$ or $\neg(\exists x\leq t)B$, then let A^* be B, $(\neg B)\wedge(\neg C)$, $(\neg B)\vee(\neg C)$, $(\exists x\leq t)(\neg B)$ or $(\forall x\leq t)(\neg B)$ respectively.

Then

$$WITNESS_A^{\vec{a}}(w,\vec{a}) \;=\; WITNESS_{A^*}^{\vec{a}}(w,\vec{a}).$$

Definition: Let $A(\vec{a})$ be a $\Sigma_1^b(PV)$-formula. We say that PV *essentially proves* $(\forall\vec{x})A(\vec{x})$ iff there is a PV-function symbol F such that

$$PV \vdash WITNESS_A^{\vec{a}}(F(\vec{a}),\vec{a}) \neq 0.$$

We shall see below that if $A \in \Sigma_1^b(PV)$ and $S_2^1(PV) \vdash (\forall\vec{x})A(\vec{x})$ then PV essentially proves $(\forall\vec{x})A(\vec{x})$.

Proposition 5: Let $A(b,\vec{a})$ be a $\Sigma_1^b(PV)$-formula and let $B(\vec{a})$ be $A(t(\vec{a}),\vec{a})$ for some term t. Then PV proves

$$WITNESS_B^{\vec{a}}(w,\vec{a}) \neq 0 \supset WITNESS_A^{b,\vec{a}}(w,t(\vec{a}),\vec{a}) \neq 0.$$

Proof: by induction on the complexity of A. □

Proposition 6: Let A be a $\Sigma_1^b(PV)$-formula and let $\vec{a}$ be a vector of free variables containing all the free variables of A. Then there are functions $MINWIT_A^{\vec{a}}$ and $WITSIZE_A^{\vec{a}}$ definable by PV such that

$$PV \vdash MINWIT_A^{\vec{a}}(v) \leq WITSIZE_A^{\vec{a}}(\vec{a})$$

and

$$PV \vdash WITNESS_A^{\vec{a}}(w,\vec{a}) \neq 0 \supset WITNESS_A^{\vec{a}}(MINWIT_A^{\vec{a}}(w),\vec{a}) \neq 0.$$

So $MINWIT_A^{\vec{a}}$ maps any witness for $A(\vec{a})$ to a minimal witness; the Gödel number of the minimal witness is bounded uniformly by $WITSIZE_A^{\vec{a}}(\vec{a})$.

The proof of Proposition 6 is by induction on the complexity of A. The crucial point of the proof is to show that sequences can be coded efficiently. For example, any sequence $<n_1, \ldots, n_m>$ has some Gödel number less than $t_m(n_1, \ldots, n_m)$ for some fixed term t_m. Although we have not specified the details of PV's β function, for any reasonable definition of the β function Proposition 6 is valid.

6.4. The Main Proof, revisited.

We next state and prove a slightly stronger version of Theorem 5.5. All the conventions of §5.2 apply here; in particular, $S_2^1(PV)$ is a natural deduction theory.

Theorem 7: Suppose $S_2^1(PV)$ proves the sequent $\Gamma,\Pi \longrightarrow \Lambda,\Delta$ and that each formula in $\Gamma\cup\Delta$ is a $\Sigma_1^b(PV)$-formula and each formula in $\Pi\cup\Lambda$ is a $\Pi_1^b(PV)$-formula. Let $c_1,\ldots,c_p$ be the free variables in $\Gamma,\Pi \longrightarrow \Lambda,\Delta$. Let X and Y be the formulae

$$X = (\bigwedge\Gamma)\wedge\bigwedge\{\neg C : C\in\Lambda\}$$

$$Y = (\bigvee\Delta)\vee\bigvee\{\neg C : C\in\Pi\}.$$

Then there is a PV-function symbol F such that

$$PV \vdash WITNESS_X^{\vec{c}}(w,\vec{c})\neq 0 \supset WITNESS_Y^{\vec{c}}(F(w,\vec{c}),\vec{c})\neq 0.$$

It is immediately obvious that Theorem 4 follows from Theorem 7, since when A is atomic,

$$PV \vdash WITNESS_A^{\vec{a}}(w,\vec{a})\neq 0 \leftrightarrow A(\vec{a}).$$

Proof: of Theorem 7.

By Proposition 4.8, there is a free cut free $S_2^1(PV)$-proof P of $\Gamma,\Pi \longrightarrow \Lambda,\Delta$ which is in free variable normal form. Every formula of P is in $\Sigma_1^b(PV)\cup\Pi_1^b(PV)$ and each cut formula of P is a $\Sigma_1^b(PV)$-formula. As in the proof of Theorem 5.5, we assume without loss of generality that Π and Λ are empty. The proof is by induction on the number of inferences in P.

To begin the proof, suppose P has no inferences. Then P contains a single sequent which must be either (a) an equality axiom, (b) a *BASIC* axiom, or (c) one of the axioms of $S_2^1(PV)$ defining an initial or defined function symbol of PV. Since all of these axioms are open and are theorems of PV, it is easy to see that the theorem holds in this case.

Now suppose that Theorem 7 holds for proofs with $\leq n$ inferences and that P has $n+1$ inferences. The argument splits into many cases depending on the last inference of P. We shall number the cases as in the proof of Theorem 5.5. Since the proof parallels closely the proof of Theorem 5.5 we shall omit a lot of the cases.

Case (1): ($\neg$:left) and ($\neg$:right). These are "cosmetic" inferences.

Case (2): ($\wedge$:left). Suppose the last inference of P is:

$$\frac{B,\Gamma^* \longrightarrow \Delta}{B\wedge C,\Gamma^* \longrightarrow \Delta}$$

Let D be the formula $B\wedge(\bigwedge\Gamma^*)$ and let E be $(B\wedge C)\wedge(\bigwedge\Gamma^*)$. The induction hypothesis is that for some PV–function symbol G,

$$PV\vdash WITNESS_D^{\vec{c}}(w,\vec{c})\neq 0\supset WITNESS_{\bigvee\Delta}^{\vec{a}}(G(w,\vec{c}),\vec{c})\neq 0.$$

Let H be the PV–function symbol defined so that $H(w)=<\beta(1,\beta(1,w)),\beta(2,w)>$ and let $F(w,\vec{c})=G(H(w),\vec{c})$. Then

$$PV\vdash WITNESS_E^{\vec{c}}(w,\vec{c})\neq 0\supset WITNESS_D^{\vec{c}}(H(w),\vec{c})\neq 0$$

and

$$PV\vdash WITNESS_E^{\vec{c}}(w,\vec{c})\neq 0\supset WITNESS_{\bigvee\Delta}^{\vec{c}}(F(w,\vec{c}),\vec{c})\neq 0$$

which is what we wanted to show.

Cases (3)-(7): Omitted.

Case (8): ($\wedge$:right). Suppose the last inference of P is:

$$\frac{\Gamma\longrightarrow B,\Delta^* \qquad\qquad \Gamma\longrightarrow C,\Delta^*}{\Gamma\longrightarrow B\wedge C,\Delta^*}$$

Let D be the formula $B\vee(\bigvee\Delta^*)$, let E be $C\vee(\bigvee\Delta^*)$ and let F be $(B\wedge C)\vee(\bigvee\Delta^*)$. The induction hypothesis is that there are PV–functions G and H such that

$$PV\vdash WITNESS_{\bigwedge\Gamma}^{\vec{c}}(w,\vec{c})\neq 0\supset WITNESS_D^{\vec{c}}(G(w,\vec{c}),\vec{c})\neq 0$$

and

$$PV\vdash WITNESS_{\bigwedge\Gamma}^{\vec{c}}(w,\vec{c})\neq 0\supset WITNESS_E^{\vec{c}}(H(w,\vec{c}),\vec{c})\neq 0.$$

Define the PV–function K so that

$$K(v,w,\vec{c}) = \begin{cases} v & \text{if } WITNESS_{\bigvee\Delta^*}^{\vec{c}}(v,\vec{c})\neq 0 \\ w & \text{otherwise} \end{cases}$$

and F so that

$$F(w,\vec{c}) \;=\; <<\beta(1,G(w,\vec{c})),\beta(1,H(w,\vec{c}))>, \\ K(\beta(2,G(w,\vec{c})),\beta(2,H(w,\vec{c})),\vec{c})>.$$

It is easy to see that

$$PV \vdash WITNESS^{\vec{c}}_{\bigwedge\Gamma}(w,\vec{c}) \neq 0 \supset WITNESS^{\vec{c}}_{F}(F(w,\vec{c}),\vec{c}) \neq 0.$$

Case (9): ($\exists\leq$:right). Omitted.

Case (10): ($\forall\leq$:right). Suppose the last inference of P is:

$$\frac{a \leq |r|, \Gamma \longrightarrow B(a), \Delta^*}{\Gamma \longrightarrow (\forall x \leq |r|)B(x), \Delta^*}$$

where a is the eigenvariable and must not appear in the lower sequent. Let D be the formula $a \leq |r| \wedge (\bigwedge\Gamma)$, let E be $B(a) \vee (\bigvee\Delta^*)$ and let C be $(\forall x \leq |r|)B(x) \vee (\bigvee\Delta^*)$. By the induction hypothesis there is a PV–function G so that

$$PV \vdash WITNESS^{\vec{c},a}_{D}(w,\vec{c},a) \neq 0 \supset WITNESS^{\vec{c},a}_{E}(G(w,\vec{c},a),\vec{c},a) \neq 0.$$

By Proposition 1, there is a PV–function H such that PV proves

$$\beta(0,H(w,\vec{c})) = |r|+1 \wedge 0 \neq (\overline{\forall} x \leq |r|)(\beta(x+1,H(w,\vec{c})) = \beta(1,G(w,\vec{c},x))).$$

We define the PV–function symbol J as follows by limited iteration on notation:

$$J(w,\vec{c},0) \;=\; \beta(2,G(w,\vec{c},0))$$

$$J(w,\vec{c},s_i(x)) \;=\; \begin{cases} J(w,\vec{c},x) & \text{if } WITNESS^{\vec{c}}_{\bigvee\Delta^*}(J(w,\vec{c},x),\vec{c}) \neq 0 \\ \beta(2,G(w,\vec{c},|s_i(x)|_d)) & \text{otherwise} \end{cases}$$

Then define $F(w,\vec{c}) = <H(w,\vec{c}),J(w,\vec{c},s_1(r(\vec{c})))>$. PV can use an induction on notation argument to prove

$$WITNESS^{\vec{c}}_{\bigwedge\Gamma}(w,\vec{c}) \neq 0 \supset WITNESS^{\vec{c}}_{C}(F(<0,w>,\vec{c}),\vec{c}) \neq 0.$$

Case (11): (*Cut*). Omitted.

Case (12): (Σ_1^b–*PIND*). Suppose the last inference of P is

$$\frac{B(\lfloor\frac{1}{2}a\rfloor),\Gamma^*\longrightarrow B(a),\Delta^*}{B(0),\Gamma^*\longrightarrow B(t),\Delta^*}$$

where the eigenvariable a must not appear in the lower sequent. We only consider the case where $B(0)$ is in Γ and $B(t)$ is in Δ.

Let D be the formula $B(\lfloor\frac{1}{2}a\rfloor)\wedge(\bigwedge\Gamma^*)$, let E be $B(a)\vee(\bigvee\Delta^*)$, let C be $B(0)\wedge(\bigwedge\Gamma^*)$ and let A be $B(t)\vee(\bigvee\Delta^*)$. By the induction hypothesis, there is a PV–function G such that

$$PV\vdash WITNESS_D^{\vec{c},a}(w,\vec{c},a)\neq 0\supset WITNESS_E^{\vec{c},a}(G(w,\vec{c},a),\vec{c},a)\neq 0.$$

Let TRM be the PV–function satisfying $TRM(z,i)=MSP(z,|z|\dot{-}i)$. Let $J(v,w)=<\beta(1,w),\beta(2,v)>$. Let H be the PV–function defined by the following limited iteration on notation:

$$H(w,\vec{c},0) = MINWIT_E^{\vec{c},a}(G(w,\vec{c},0))$$

$$H(w,\vec{c},s_i(x)) = \begin{cases} H(w,\vec{c},x) & \text{if } WITNESS_{\bigvee\Delta^*}^{\vec{c}}(\beta(2,H(w,\vec{c},x)),\vec{c})\neq 0 \\ & \text{or if } |x|_d\geq|t(\vec{c})| \\ MINWIT_E^{\vec{c},a}(G(J(w,H(w,\vec{c},x)),\vec{c},TRM(t(\vec{c}),|s_i(x)|_d))) & \\ & \text{otherwise} \end{cases}$$

This is a valid limited iteration on notation definition since

$$PV\vdash H(w,\vec{c},x)\leq WITSIZE_E^{\vec{c},a}(\vec{c},a)$$

because $MINWIT_E^{\vec{c},a}$ was used in the definition of H. By using induction on notation, PV can prove

$$|x|_d\leq|t(\vec{c})|\wedge WITNESS_C^{\vec{c}}(w,\vec{c})\neq 0\supset WITNESS_E^{\vec{c},a}(H(w,\vec{c},x),\vec{c},TRM(t(\vec{c}),|x|_d))\neq 0.$$

So define $F(w,\vec{c})=H(w,\vec{c},s_1(t(\vec{c})))$ and then

$$PV\vdash WITNESS_C^{\vec{c}}(w,\vec{c})\neq 0\supset WITNESS_A^{\vec{c}}(F(w,\vec{c}),\vec{c})\neq 0.$$

Q.E.D. □

Corollary 8: Let $A(\vec{a})$ be a $\Sigma_1^b(PV)$–formula. If $S_2^1(PV) \vdash (\forall\vec{x})A(\vec{x})$ then PV essentially proves $(\forall\vec{x})A(\vec{x})$.

Proof: This is immediate from the definition of "essentially proves" and Theorem 7. □

Chapter 7

Gödel Incompleteness Theorems

We next take up the subject of Gödel incompleteness results. We shall see that the first and second incompleteness theorems hold for S_2^1. We shall also prove strengthened versions of the incompleteness theorems which apply to the consistency of bounded proofs and of free cut free proofs.

Before proving incompleteness results, we must show that the syntax of metamathematics can be coded in S_2^1. Of course, it is well known that the syntax of first-order logic can be recognized and manipulated by polynomial time algorithms and as we showed earlier, S_2^1 can Σ_1^b-define any polynomial time algorithm. This might appear to be an a priori argument that the arithmetization of metamathematics can be carried out in S_2^1. However, as Feferman [9] emphasizes, the arithmetization of metamathematics must be carried out in an intensional manner and this does not follow from our a priori argument.

We begin by giving a general framework for making inductive definitions in S_2^1 and using this framework to outline how the arithmetization of metamathematics in S_2^1 can be carried out intensionally.

7.1. Trees.

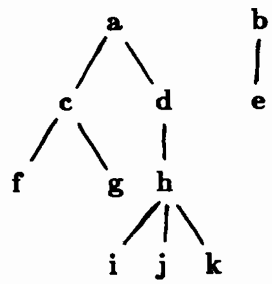
As a preliminary we need to give a method for coding trees in S_2^1. Trees will be coded by sequences. An example of a tree and its coding are given in Figure 2. A tree is coded by a sequence with two special symbols "[" and "]" for denoting the structure of the tree.

Following the notations and conventions of §2.4-2.5 we define the following Σ_1^b-definable functions and Δ_1^b-predicates of S_2^1.

(a) $$\begin{aligned} &\mathit{RBracket} = 0 \\ &\mathit{LBracket} = 1 \\ &\mathit{Node}(x) \iff x \geq 2 \end{aligned}$$

(b) $$\begin{aligned} &\mathit{Balanced}(w) \iff \\ &\quad\iff [(\#j<\mathit{Len}(w))(\mathit{LBracket}=\beta(Sj,w)) = (\#j<\mathit{Len}(w))(\mathit{RBracket}=\beta(Sj,w))] \wedge \\ &\quad\quad \wedge (\forall i<\mathit{Len}(w))[(\#j\leq i)(\mathit{RBracket}=\beta(Sj,w)) \leq (\#j\leq i)(\mathit{LBracket}=\beta(Sj,w))] \end{aligned}$$

Note that the counting operations are all equivalent to length bounded counting and hence by Theorem 2.7, *Balanced* is a Δ_1^b-definable predicate. We shall use length bounded counting freely without comment from now on.



a[c[fg]d[h[ijk]]]b[e]

A tree is coded by a sequence which enumerates the tree in depth first order. Two special symbols, "[" and "]" are used to denote movement down and up the tree. The tree shown has two roots, labeled **a** and **b**.

Figure 2

(c) $Depth(i,w) \;=\; (\#j<i)(LBracket{=}\beta(Sj,w)) \dot{-} (\#j<i)(RBracket{=}\beta(Sj,w))$

(d) $$MultiTree(w) \iff Seq(w) \wedge Len(w) \neq 0 \wedge Balanced(w) \wedge LBracket \neq \beta(1,w) \wedge \wedge (\forall i<Len(w))(LBracket=\beta(i+1,w) \supset Node(\beta(i+2,w)))$$

MultiTree(w) is true iff w codes a tree, which may have more than one root.

(e) $Tree(w) \iff MultiTree(w) \wedge \neg(\exists i<Len(w))(i>0 \wedge Node(\beta(Sj,w)) \wedge Depth(Sj,w)=0)$

(f) $Leaf(i,w) \iff MultiTree(w) \wedge Node(\beta(i,w)) \wedge LBracket \neq \beta(Si,w)$

So $Leaf(i,w)$ is true iff $\beta(i,w)$ codes a leaf of the tree w. The *father* of a node is the node directly above it; the *sons* of a node are the nodes directly below it. We define *Father* and *Son* so that if $Father(i,w)=k$ then the node $\beta(k,w)$ is the father of the node $\beta(i,w)$, and so that $Son(n,j,w)=i$ iff $\beta(i,w)$ is the n-th son of the node $\beta(j,w)$.

(g) $$Father(i,w) = \begin{cases} (\mu j<i)Balanced(Subseq(w,j+2,i)) & \text{if } Node(i,w) \\ Len(w)+1 & \text{otherwise} \end{cases}$$

We use $Len(w)+1$ as the alternative value for the function *Father* since $Len(w)+1$ is never a node and hence never a valid father.

(h) $$SonPos(i,j,w) = \begin{cases} 0 & \text{if } Father(i,w)\neq j \\ (\#z\leq i \dot{-} j)[Father(j+z,w)=j] & \text{otherwise} \end{cases}$$

$$Son(k,j,w) \;=\; (\mu i \le Len(w))(SonPos(i,j,w)=k)$$

Note that the father of a root of a multitree is 0 and that the roots of a multitree are the sons of an imaginary node at the zeroth position of the sequence coding the multitree.

(i) $Valence\,(j,w) \;=\; (\#z \le Len(w))(Father(z,w)=j)$

(j) $Son\beta(k,j,w) \;=\; \beta(Son(k,j,w),w) \dot{-} 2$

$$Father\beta(i,w) \;=\; \beta(Father(i,w),w) \dot{-} 2$$

$$Root\beta(w) \;=\; \beta(1,w) \dot{-} 2$$

$$Node\beta(i,w) \;=\; \beta(i,w) \dot{-} 2$$

We subtract 2 so that the values of the node labels are distinct from the codes for brackets, namely 0 and 1 for "[" and "]".

(k) We also define a function for extracting subtrees of trees:

$$SubTree(i,w) \;=\; Subseq(i, \max\{j \le Len(w)+1 : Tree(Subseq(i,j,w))\}, w).$$

The above encoding of trees is intensional in the sense of Feferman [9]. The skeptical reader may verify that, for instance, S_2^1 can prove

$$MultiTree(w) \land Node(\beta(j,w)) \supset (Leaf(j,w) \leftrightarrow Valence\,(j,w)=0)$$

$$MultiTree(w) \land Node(\beta(i,w)) \supset Depth(Father(i,w),w)=Depth(i,w) \dot{-} 1$$

$$MultiTree(w) \land Node(i,w) \supset Tree(SubTree(i,w)).$$

7.2. Inductive Definitions.

We show in this section that S_2^1 is capable of defining predicates and functions by inductive definitions, provided that the inductive definitions give a straightforward deterministic polynomial time algorithm for expanding the inductive definition. Theorem 2 shows that such an inductive definition is intensional and allows proofs in S_2^1 to be carried out by induction on the complexity of an inductive definition. We later use the constructions of this section to argue that S_2^1 can arithmetize metamathematics.

Definition: The n predicates $P_0, \ldots, P_{n-1}$ are defined by a *p-inductive definition* iff they are defined by the following:

(a) k is a non-negative integer,

(b) For each $s \leq k$ there is a number $i_s \geq 0$ and a formula Q_s:

$$Q_s(x) \iff R_{s,0}(f_{s,0}(x)) \wedge \cdots \wedge R_{s,i_s}(f_{s,i_s}(x))$$

where the following conditions hold:

(i) each $R_{s,j}$ is P_i or $\neg P_i$ for some $i<n$,

(ii) each $f_{s,j}$ is a Σ_1^b–definable function of S_2^1,

(iii) for each $j \leq i_s$, $S_2^1 \vdash x \neq 0 \supset |f_{s,j}(x)| < |x|$,

(iv) $S_2^1 \vdash |f_{s,0}(x)| + \cdots + |f_{s,i_s}(x)| \leq |x|$.

(c) For each $i<n$ there is a function g_i which is Σ_1^b–definable in S_2^1 such that $S_2^1 \vdash (\forall x)(g_i(x) \leq k)$.

(d) For each $i<n$, either $P_i(0)$ or $\neg P_i(0)$ is true by explicit definition.

(e) For each $i<n$, P_i is inductively defined by

$$P_i(x) \iff Q_{g_i(x)}(x).$$

Because of the decreasing length condition of (b.iii) above, a p-inductive definition uniquely defines the value of $P_i(x)$ for all i and x. In fact, a p-inductive definition gives a polynomial time deterministic algorithm for checking whether $P_i(x)$ holds. That this algorithm can be formalized in S_2^1 is the content of the next theorem.

Theorem 1: Let $P_0, \ldots, P_{n-1}$ be defined as in the p-inductive definition above. Then each predicate P_i is Δ_1^b–definable in S_2^1.

Proof: In order to Δ_1^b–define the predicates P_i in S_2^1 we must (at least implicitly) specify an algorithm for determining when $P_i(x)$ holds. This is done by constructing a tree which demonstrates that either $P_i(x)$ or $\neg P_i(x)$ holds. Each node of the tree will be labeled as $<P_m, y>$ or $<\neg P_m, y>$ which denote the assertions that $P_m(y)$ or $\neg P_m(y)$ holds, respectively. The sons of such a node must provide evidence that $P_m(y)$ or $\neg P_m(y)$ (respectively) is valid. For example, if $g_m(y)=s$, then the sons of $<P_m, y>$ must be $1+i_s$ nodes labeled $<R_{s,j}, f_{s,j}(y)>$ for $j \leq i_s$. The leaves of the tree must be labeled either $<P_m, 0>$ or $<\neg P_m, 0>$ as allowed by clause (c) of the p-inductive definition. The root of the tree will be labeled either $<P_i, x>$ or $<\neg P_i, x>$.

We begin by writing out a formal definition for a "demonstration tree" for $P_i(x)$. Let $C_{s,j}$, $\overline{C}_{s,j}$, $B_{s,j}$, and D_i be fixed terms defined by:

$$C_{s,j} = \begin{cases} I_i & \text{if } R_{s,j} \text{ is } P_i \\ I_{i+n} & \text{if } R_{s,j} \text{ is } \neg P_i \end{cases}$$

$$\overline{C}_{s,j} = \begin{cases} I_i & \text{if } R_{s,j} \text{ is } \neg P_i \\ I_{i+n} & \text{if } R_{s,j} \text{ is } P_i \end{cases}$$

$$B_{s,j} = I_i \qquad \text{where } R_{s,j} \text{ is } \neg P_i \text{ or } P_i$$

$$D_i = \begin{cases} I_i & \text{if } P_i(0) \\ I_{i+n} & \text{if } \neg P_i(0) \end{cases}$$

(Recall that I_j is a term with value j.) The leaves of a "demonstration tree" must satisfy the leaf condition:

$$DTLC(u,w) = (\bigvee_{i=0}^{n-1} \beta(1,Node\beta(u,w)){=}D_i) \wedge \beta(2,Node\beta(u,w)){=}0$$

or, in words, a leaf must be labeled $<D_i,0>$ for some i. The non-leaf nodes of the "demonstration tree" must satisfy the following condition:

$$\begin{aligned} DTNC2(u,w) \iff & \beta(1,Node\beta(u,w)){<}2n \wedge \beta(2,Node\beta(u,w)){\neq}0 \wedge \\ & \wedge \bigwedge_{s=0}^{k} \bigwedge_{i=0}^{n-1} [Rem(\beta(1,Node\beta(u,w)),n){=}i \wedge g_i(\beta(2,Node\beta(u,w))){=}s \supset \\ & \quad \supset Valence\,(u,w){=}i_s{+}1 \wedge \bigwedge_{j=0}^{i_s} f_{s,j}(\beta(2,Node\beta(u,w))){=}\beta(2,Son\beta(Sj,u,w)) \wedge \\ & \quad \wedge(\beta(1,Node\beta(u,w)){<}n \supset \bigwedge_{j=0}^{i_s} C_{s,j}{=}\beta(1,Son\beta(Sj,u,w))) \wedge \\ & \quad \wedge(\beta(1,Node\beta(u,w)){\geq}n \supset \bigvee_{j=0}^{i_s} \overline{C}_{s,j}{=}\beta(1,Son\beta(Sj,u,w)))]. \end{aligned}$$

We combine both these requirements in

$$DTNC1(u,w) \iff (Leaf(u,w) \supset DTLC(u,w)) \wedge (\neg Leaf(u,w) \supset DTNC2(u,w)).$$

So a "demonstration tree" which proves $P_i(x)$ or $\neg P_i(x)$ must satisfy

$$\begin{aligned} DemoTree_i(w,x) \iff & Tree(w) \wedge (\forall u{<}Len(w))(Node(Su,w) \supset DTNC1(Su,w)) \wedge \\ & \wedge(Root\beta(w){=}{<}i,x{>} \vee Root\beta(w){=}{<}i{+}n,x{>}). \end{aligned}$$

We will introduce P_i in S_2^1 as a Δ_1^b–defined predicate symbol by:

$$\begin{aligned} P_i(x) \iff & (\exists w)(DemoTree_i(w,x) \wedge \beta(1,Root\beta(w)){=}i) \\ \iff & \neg(\exists w)(DemoTree_i(w,x) \wedge \beta(1,Root\beta(w)){=}i{+}n) \end{aligned}$$

Thus it will suffice to establish that S_2^1 proves

$$(\forall x)(\exists w)DemoTree_i(w,x)$$

and

$$DemoTree_i(w,x)\wedge DemoTree_i(v,x)\supset\beta(1,Root\beta(w))=\beta(1,Root\beta(v)).$$

Since it is easier, we first show that S_2^1 proves the uniqueness condition. We argue informally inside the theory S_2^1. Suppose w and v are $DemoTree$'s for $P_i(x)$ and/or $\neg P_i(x)$. Let $A(w,v,b)$ be the formula

$$(\forall u<b)[(\neg Node(Su,w)\supset\beta(Su,w)=\beta(Su,v))\wedge \\ \wedge(Node(Su,w)\supset\beta(2,Node\beta(Su,w))=\beta(2,Node\beta(Su,v)))].$$

It follows from the definition of $DemoTree$ that $A(w,v,b)\supset A(w,v,Sb)$. Hence, by Σ_1^b-$LIND$, $A(w,v,Len(v))$ and $A(w,v,Len(w))$ and hence $Len(w)=Len(v)$. Now let $B(w,v,b)$ be the formula

$$(\forall u<Len(w))[u\geq Len(w)\dot{-}b\wedge Node(Su,w)\supset\beta(1,Node\beta(Su,w))=\beta(1,Node\beta(Su,v))].$$

Now it follows that $B(w,v,b)\supset B(w,v,Sb)$ so by Σ_1^b-$LIND$, $B(w,v,Len(w))$. But this immediately tells us that

$$\beta(1,Root\beta(w))=\beta(1,Root\beta(v)).$$

This completes the S_2^1-proof of the uniqueness condition.

The rest of the proof of Theorem 1 is devoted to establishing the existence condition for $DemoTree_i$. It is tempting to just argue by induction on the length of x that a $DemoTree_i$ exists. Unfortunately, this argument would use Π_2^b-$PIND$ and we can not carry this out in S_2^1. Instead we must use a more sophisticated argument to construct the $DemoTree$. What we will do is formalize a breadth first algorithm which constructs the demonstration tree and then labels the nodes appropriately. We first define:

$$\begin{aligned} PDTNC2(u,w) \iff{} & \beta(1,Node\beta(u,w))<n\wedge\beta(2,Node\beta(u,w))\neq 0\wedge \\ & \wedge\bigwedge_{s=0}^{k}\bigwedge_{i=0}^{n-1}[\beta(1,Node\beta(u,w))=i\wedge g_i(\beta(2,Node\beta(u,w)))=s\supset \\ & \supset Valence\,(u,w)=i_s+1\wedge\bigwedge_{j=0}^{i_s}f_{s,j}(\beta(2,Node\beta(u,w)))=\beta(2,Son\beta(Sj,u,w))\wedge \\ & \wedge\bigwedge_{j=0}^{i_s}B_{s,j}=\beta(1,Son\beta(Sj,u,w))] \end{aligned}$$

$$PDTLC(u,w) \iff \beta(1,Node\beta(u,w))<n\wedge\beta(2,Node\beta(u,w))=0$$

$$PDTNC1(u,w) \iff (Leaf(u,w) \supset PDTLC(u,w)) \wedge (\neg Leaf(u,w) \supset PDTNC2(u,w))$$

$$\begin{aligned} PDT_i(w,x,b) \iff & Tree(w) \wedge Root\beta(w) = <i,x> \wedge \\ & \wedge(\forall u < Len(w))(Node(Su,w) \wedge Depth(Su,w) < b \supset PDTNC1(Su,w)) \wedge \\ & \wedge(\forall u < Len(w))(Node(Su,w) \supset |\beta(2, Node\beta(Su,w))| \leq |x| \dot{-} Depth(Su,w)) \wedge \\ & \wedge(SizeBounds) \end{aligned}$$

where we explain *SizeBounds* below. So $PDT_i(w,x,b)$ asserts that w is a tree containing the first $b+1$ levels of the construction of a demonstration tree. The *SizeBounds* is a formula which bounds the size of w. What we wish to show is

$$S_2^1 \vdash (\exists w \leq t(x,b))(PDT_i(w,x,b)) \supset (\exists w \leq t(x,Sb))(PDT_i(w,x,Sb))$$

for some term t; the *SizeBounds* formula must contain enough information to do this. We argue informally how $t(x,b)$ may be found. First, we count the non-leaf nodes u, which are labeled $<i,y>$ with $y \neq 0$. The number of bits used to code such a node can be required to be not more than $2(2 \cdot |i| + 2 \cdot |y| + 4) + 2$ which is $\leq 4 \cdot |n| + 4 \cdot |y| + 10$. We add on an adjustment allowing for the bits needed to code two brackets and conclude that each node can be coded by $\leq 4 \cdot |n| + 18 + 4 \cdot |y|$ bits. Consider the non-leaf nodes which are of depth $c \leq b$; there are at most $|x|$ of them and their total length is $\leq |x|$. Hence the total number of bits used to code the nodes at depth c is bounded by

$$\begin{aligned} \sum_{Depth(<i,y>)=c} (4 \cdot |y| + 4 \cdot |n| + 18) \ &\leq \ 4 \textstyle\sum(|y|) + |x| \cdot (4 \cdot |n| + 18) \\ &\leq \ |x| \cdot (4 + 4 \cdot |n| + 18) \ \leq \ |x| \cdot (4 \cdot |n| + 22). \end{aligned}$$

Since the tree has depth b, the total number of bits required to code the non-leaf nodes of the tree is $\leq (b+1) \cdot |x| \cdot (4 \cdot |n| + 22)$.

We must also consider the nodes labeled $<i,0>$. Let $i_{\max}$ be $\max\{i_s : s=0, \ldots, k\}$. There are $\leq b \cdot |x|$ non-leaf nodes on the first b levels of the tree and below them are $\leq 1 + b \cdot |x| \cdot (i_{\max} + 1)$ leaf nodes. (The extra 1 is for the case $x=0$). Since a label $<i,0>$ and its surrounding brackets can be coded by $\leq 4 \cdot |n| + 18$ bits, the total number of bits used to code these nodes is bounded by

$$(1 + b \cdot |x| \cdot (1 + i_{\max}))(4 \cdot |n| + 18).$$

So the length $|w|$ of w is bounded by

$$s(x,b) \ = \ (1 + |x| \cdot (1 + b \cdot (2 + i_{\max}))) \cdot (4 \cdot |n| + 22).$$

Since b will be restricted to be $\leq |x|$ we can define the term $t(x,b)$ to be equal to $2^{s(x,b)}$. This is the desired bound on w.

The formula *SizeBounds* should be a formula containing all of the information used above in establishing the bound on $|w|$. It is rather complicated to actually write out *SizeBounds*, so we leave it as an exercise for the skeptical reader. Given that *SizeBounds* is properly formulated, it is now straightforward for S_2^1 to prove

$$(\exists w \le t(x,b))PDT_i(w,x,b) \supset (\exists x \le t(x,Sb))PDT_i(w,x,Sb).$$

So by Σ_1^b-*LIND* , $S_2^1 \vdash (\exists w \le t(x,|x|))PDT_i(w,x,|x|)$. Finally, we need to show that

$$S_2^1 \vdash (\exists w)PDT_i(w,x,|x|) \supset (\exists w)DemoTree_i(w,x).$$

So let $C(w,v,b)$ be the formula

$$\begin{aligned} &Len(w)=Len(v) \wedge (\forall u < Len(w))[(u < Len(w) \dot{-} b \supset \beta(Su,w)=\beta(Su,v)) \wedge \\ &\quad \wedge (u \ge Len(w) \dot{-} b \wedge \neg Node(Su,w) \supset \beta(S\,u,w)=\beta(Su,v)) \wedge \\ &\quad \wedge (u \ge Len(w) \dot{-} b \wedge Node(Su,w) \supset DTNC1(Su,v) \wedge \\ &\qquad \wedge \beta(2,Node\beta(Su,w))=\beta(2,Node\beta(Su,v)) \wedge \\ &\qquad \wedge \beta(1,Node\beta(Su,w))=Rem(\beta(1,Node\beta(Su,v)),n))]. \end{aligned}$$

It is quite easy to see that $S_2^1 \vdash PDT_i(w,x,|x|) \supset (\exists v \le w)C(w,v,0)$ and

$$S_2^1 \vdash PDT_i(w,x,|x|) \wedge (\exists v \le w \cdot 2^{2 \cdot |n| \cdot b})C(w,v,b) \supset (\exists x \le w \cdot 2^{2 \cdot |n| \cdot (b+1)})C(w,v,b+1).$$

Hence, by Σ_1^b-*LIND* ,

$$S_2^1 \vdash PDT_i(w,x,|x|) \supset (\exists v \le w \cdot 2^{2 \cdot |n| \cdot Len(w)})C(w,v,Len(w))$$

from which $S_2^1 \vdash (\exists v)DemoTree_i(v,x)$ is immediate.

Q.E.D. □

Theorem 1 states that S_2^1 can Δ_1^b-define predicates which have p-inductive definitions. We also want S_2^1 to be able to prove theorems involving p-inductively defined predicates. Accordingly we need to know that certain kinds of inductive proofs can be formalized in S_2^1.

Definition: Let $P_0, \ldots, P_{n-1}$ be defined p-inductively as above. We say that the $2n$ formulae

$$(\forall x)(P_i(x) \supset B_i(x))$$

$$(\forall x)(\neg P_i(x) \supset C_i(x))$$

have a *p-inductive proof* iff the following hold:

(a) Each B_i and C_i is Δ_1^b with respect to S_2^1.

(b) For $0 \leq s \leq k$, $0 \leq j \leq i_s$, $R_{s,j}$ is as in clause (b) of the p-inductive definition for $P_0, \ldots, P_{n-1}$. Let $Q_{s,j}$ be the formula B_i if $R_{s,j} = P_i$ or the formula C_i if $R_{s,j} = \neg P_i$. Define $\overline{Q}_{s,j}$ dually to be B_i if $R_{s,j} = \neg P_i$ and to be C_i if $R_{s,j} = P_i$.

(c) For $i = 0, \ldots, n-1$, and $0 \leq s \leq k$, S_2^1 proves

$$g_i(x) = s \wedge \left(\bigwedge_{j=0}^{i_s} (R_{s,j}(f_{s,j}(x)) \wedge Q_{s,j}(f_{s,j}(x)))\right) \supset B_i(x)$$

$$g_i(x) = s \wedge \left(\bigvee_{j=0}^{i_s} (\neg R_{s,j}(f_{s,j}(x)) \wedge \overline{Q}_{s,j}(f_{s,j}(x)))\right) \supset C_i(x).$$

Theorem 2: Given (a), (b), (c) as above, S_2^1 proves

$$(\forall x)\,[(P_i(x) \supset B_i(x)) \wedge (\neg P_i(x) \supset C_i(x))]$$

for $0 \leq i < n$.

Proof: Let $A(w,a,b)$ be the formula

$$\begin{aligned} &DemoTree_i(w,a) \supset (\forall u < Len(w))(u \geq Len(w) \dot{-} b \wedge Node(Su,w) \supset \\ &\quad \supset \bigwedge_{i=0}^{n-1} [\beta(1, Node\beta(Su,w)) = i \supset B_i(\beta(2, Node\beta(Su,w)))] \wedge \\ &\quad\quad \wedge \bigwedge_{i=0}^{n-1} [\beta(1, Node\beta(Su,w)) = i+n \supset C_i(\beta(2, Node\beta(Su,w)))]). \end{aligned}$$

Clearly, $S_2^1 \vdash A(w,a,0)$. Also, because of clause (c) of the p-inductive proof, S_2^1 proves $A(w,a,b) \supset A(w,a,Sb)$. Hence, by Σ_1^b-$LIND$, $S_2^1 \vdash A(w,a,Len(w))$. By Theorem 1, $S_2^1 \vdash (\exists w) DemoTree_i(w,a)$ and since the root node of such a demonstration tree must be $<i,a>$ or $<i+n,a>$ we have the desired result.

Q.E.D. □

Definition: The function F is defined by a *p-inductive definition* iff F is defined by the following:

(a) k is a fixed nonnegative integer.

(b) For each $s \le k$ there is an i_s-ary function G_s and i_s unary functions $f_{s,1}, \ldots, f_{s,i_s}$ satisfying

(i) G_s and each $f_{s,j}$ are Σ_1^b-definable functions of S_2^1.

(ii) $S_2^1 \vdash x \ne 0 \supset |f_{s,j}(x)| < |x|$ for all $j \le i_s$.

(iii) $S_2^1 \vdash |f_{s,1}(x)| + \cdots + |f_{s,i_s}(x)| \le |x|$.

(c) There is a function g Σ_1^b-definable by S_2^1 so that $S_2^1 \vdash (\forall x)(g(x) \le k)$.

(d) $F(x)$ is defined inductively by

$$F(x) = G_{g(x)}(F(f_{g(x),1}(x)), \ldots, F(f_{g(x),i_{g(x)}}(x))).$$

(e) There is a term $t(x)$ (which will bound $F(x)$) so that for all $s \le k$,

$$S_2^1 \vdash [\bigwedge_{j=1}^{i_s} a_j \le t(f_{s,j}(x))] \supset G_s(a_1, \ldots, a_{i_s}) \le t(x).$$

Theorem 3: Let F be defined by the p-inductive definition above. Then F is Σ_1^b-definable in S_2^1. Furthermore, the definition of F in S_2^1 is intensionally correct in that properties of F can be proved in S_2^1 by the use of induction.

Proof: This is proved in a manner very similar to the proofs of Theorems 1 and 2, and we omit the proof. □

7.3. The Arithmetization of Metamathematics.

In order to establish the Gödel incompleteness theorems for Bounded Arithmetic, we need to introduce Σ_1^b-defined function symbols and Δ_1^b-defined predicate symbols for handling Gödel numberings for metamathematical concepts such as "formula", "proof", etc. With the aid of p-inductive definitions we demonstrate such an arithmetization below.

We begin by introducing Gödel numbers for all the syntactic symbols of Bounded Arithmetic. Each symbol is assigned a number as listed below.

Logical Symbols

$\forall$ - 0	$\vee$ - 5
$\exists$ - 1	(- 6
$\neg$ - 2	) - 7
$\supset$ - 3	, - 8
$\wedge$ - 4	$\longrightarrow$ - 9

Non-logical symbols

Constants:	0 - 16		
Unary Functions:	S - 20,	$\lvert x \rvert$ - 24,	$\lfloor \frac{1}{2}x \rfloor$ - 28
Binary Functions:	$+$ - 32,	$\cdot$ - 36,	# - 40
Binary Relations:	$=$ - 18,	$\leq$ - 22	

Free Variables		Bound Variables	
a_1 -	19	x_1 -	17
a_2 -	23	x_2 -	21
a_3 -	27	x_3 -	25
$\vdots$	$\vdots$	$\vdots$	$\vdots$

Corresponding to this assignment of Gödel numbers we introduce the following predicate symbols in S_2^1:

$$\begin{array}{lll lll}
AQuant(x) & \Longleftrightarrow & x{=}0 & Zero(x) & \Longleftrightarrow & x{=}16 \\
EQuant(x) & \Longleftrightarrow & x{=}1 & Succ(x) & \Longleftrightarrow & x{=}20 \\
Not(x) & \Longleftrightarrow & x{=}2 & Log2(x) & \Longleftrightarrow & x{=}24 \\
Implies(x) & \Longleftrightarrow & x{=}3 & Div2(x) & \Longleftrightarrow & x{=}28 \\
And(x) & \Longleftrightarrow & x{=}4 & Plus(x) & \Longleftrightarrow & x{=}32 \\
Or(x) & \Longleftrightarrow & x{=}5 & Times(x) & \Longleftrightarrow & x{=}36 \\
LParen(x) & \Longleftrightarrow & x{=}6 & Smash(x) & \Longleftrightarrow & x{=}40 \\
RParen(x) & \Longleftrightarrow & x{=}7 & Equals(x) & \Longleftrightarrow & x{=}18 \\
Separ(x) & \Longleftrightarrow & x{=}8 & LE(x) & \Longleftrightarrow & x{=}22 \\
Arrow(x) & \Longleftrightarrow & x{=}9 & & &
\end{array}$$

$$\begin{array}{lll}
Quant(x) & \Longleftrightarrow & AQuant(x) \vee EQuant(x) \\
Conn\,2(x) & \Longleftrightarrow & Implies(x) \vee And(x) \vee Or(x) \\
Func1(x) & \Longleftrightarrow & Succ(x) \vee Log2(x) \vee Div2(x) \\
Func2(x) & \Longleftrightarrow & Plus(x) \vee Times(x) \vee Smash(x) \\
Rel2(x) & \Longleftrightarrow & Equals(x) \vee LE(x) \\
FVar(x) & \Longleftrightarrow & x > 16 \wedge Rem(x,4)=3 \\
BVar(x) & \Longleftrightarrow & x > 16 \wedge Rem(x,4)=1 \\
Var(x) & \Longleftrightarrow & FVar(x) \vee BVar(x)
\end{array}$$

(Note that we used *"Separ"* since *"Comma"* has already been used.) We will abbreviate constants by using a bar over the name of the constant. For example, $\overline{AQuant}$ denotes the constant 0, $\overline{LParen}$ denotes 6, and $\overline{LE}$ denotes 22.

Definition: *Semiterm* and *Term* are unary predicates which are Δ_1^b–defined in S_2^1 by the following inductive definition:

(a) $\neg Semiterm(0)$

(b) If $Seq(w)$ and $Len(w)=1$ and $Var(\beta(1,w)) \vee Zero(\beta(1,w))$ then $Semiterm(w)$.

(c) If $Semiterm(w)$ and $Func1(x)$ then $Semiterm((0*\overline{LParen}*x)**(w*\overline{RParen}))$.

(d) If $Semiterm(w)$, $Semiterm(v)$ and $Func2(x)$ then

$$Semiterm((0*\overline{LParen})**(w*x)**(v*\overline{RParen})).$$

(e) Anything which is not required to be a semiterm by the above conditions is not a semiterm.

It is easy to see that the definition of semiterm can be formulated as a p-inductive definition.

A term is defined to be a semiterm without any bound variables:

$$Term(w) \iff Semiterm(w) \wedge (\forall x < Len(w))(\neg BVar(\beta(Sx,w)))$$

We next define semiformulae and formulae. We shall adopt conventions on free and bound variables which are slightly unusual but which make the inductive definitions more manageable. We first define atomic formulae and atomic semiformulae by:

$$\begin{aligned} ASemiFmla(w) \iff & LParen(\beta(1,w)) \wedge RParen(\beta(Len(w),w)) \wedge (\exists x < Len(w))[Rel2(\beta(x,w)) \wedge \\ & \wedge Semiterm(Subseq(w,2,x)) \wedge Semiterm(Subseq(w,x+1,Len(w)))] \end{aligned}$$

$$AFmla(w) \iff ASemiFmla(w) \wedge (\forall x < Len(w))(\neg BVar(\beta(Sx,w)))$$

We also define what it means for a bound variable to appear bound in a semiformula:

$$Free(x,w) \iff BVar(x) \wedge Seq(w) \wedge (\forall i < Len(w) \dot{-} 1)(Quant(\beta(Si,w)) \supset x \neq \beta(i+2,w))$$

$$Bound(x,w) \iff BVar(x) \wedge Seq(w) \wedge \neg Free(x,w)$$

$$\begin{aligned} Compat(w,v) \iff & Seq(v) \wedge Seq(w) \wedge (\forall x < Len(v))(\neg Bound(\beta(Sx,v),w)) \wedge \\ & \wedge (\forall x < Len(w))(\neg Bound(\beta(Sx,w),v)) \end{aligned}$$

Because of the way we have defined *Bound* and *Free* we will not allow semiformulae in which a bound variable is both bound and free. For example, $(\forall x)(x\neq 0)\supset x\neq 0$ is not a valid semiformula.

We define $SemiFmla\,(w)$ by the following inductive definition:

(a) If $ASemiFmla(w)$ then $SemiFmla\,(w)$.

(b) If $SemiFmla\,(v)$ then $SemiFmla\,((0*\overline{LParen}*\overline{Not})**(v*\overline{RParen}))$.

(c) If $SemiFmla\,(v_1)$, $SemiFmla\,(v_2)$ and $Conn\,2(x)$ and if $Compat(v_1,v_2)$ then

$$SemiFmla\,((0*\overline{LParen})**(v_1*x)**(v_2*\overline{RParen})).$$

(d) If $SemiFmla\,(v_1)$, $Quant(x)$, $BVar(y)$, $Semiterm(v_2)$, $Free(y,v_1)$ and $Compat(v_1,v_2)$ and $(\forall u<Len(v_2))(\beta(Su,v_2)\neq y)$ then

$$SemiFmla\,((0*\overline{LParen}*\overline{LParen}*x*y*\overline{LE})**(v_2*\overline{RParen})**(v_1*\overline{RParen}))$$

and

$$SemiFmla\,((0*\overline{LParen}*\overline{LParen}*x*y*\overline{RParen})**(v_1*\overline{RParen})).$$

(e) $SemiFmla\,(w)$ is true only as required by the above clauses.

We define $Fmla(w)$ to mean that w codes a formula; that is to say, w is a semiformula and no bound variable appears free in w:

$$Fmla(w) \iff SemiFmla\,(w)\wedge(\forall u<Len(w))(BVar(\beta(Su,w))\supset Bound(\beta(Su\,,w),w))$$

We next define how to count the alternation of bounded quantifiers in a formula. This allows S_2^1 to recognize Σ_i^b–formulae. We first must be able to distinguish sharply bounded from non-sharply bounded quantifiers. We define $LTerm\,(x)$ to be true iff x codes a term of the form $|t|$:

$$LTerm\,(x) \iff Semiterm(x)\wedge Len(x)>1\wedge Log2(\beta(2,x)).$$

$QCount(w)$ is a function classifying the formula w by its alternation of quantifiers. $QCount(w)=<0,i>$ means $w\in\Sigma_i^b$, $QCount(w)=<1,i>$ means $w\in\Pi_i^b$, and $QCount(w)=<2,i>$ means $w\in\Sigma_i^b\cap\Pi_i^b$. $QCount$ is defined by the following p-inductive definition:

(a) If $\neg SemiFmla\,(w)$ then $QCount(w)=0$.

(b) If $ASemiFmla(w)$ then $QCount(w)=<2,0>$.

(c) If $w=(0*\overline{LParen}*\overline{Not})**(v*\overline{RParen})$ then

$$QCount(w) = \begin{cases} <0,i> & \text{if } QCount(v)=<1,i> \\ <1,i> & \text{if } QCount(v)=<0,i> \\ <2,i> & \text{if } QCount(v)=<2,i> \\ 0 & \text{otherwise} \end{cases}$$

(d) Suppose $w=(0*\overline{LParen})**(v_1*x)**(v_2*RParen)$, where $Conn\,2(x)$, $SemiFmla\,(v_1)$, $SemiFmla\,(v_2)$ and $Compat(v_1,v_2)$. If $QCount(v_1)=0$ or $QCount(v_2)=0$ then $QCount(w)=0$. Otherwise, define

$$QImp(v_1) = \begin{cases} QCount(v_1) & \text{if } \neg Implies(x)\vee\beta(1,QCount(v_1))=2 \\ <1\dot{-}\beta(1,QCount(v_1)),\beta(2,QCount(v_1))> & \text{otherwise} \end{cases}$$

and let i_1,j_1,i_2,j_2 be so that $QImp(v_1)=<i_1,j_1>$ and $QCount(v_2)=<i_2,j_2>$. Then

$$QCount(w) = \begin{cases} <i_1,j_1> & \text{if } j_2<j_1\vee(i_2=2\wedge j_1=j_2) \\ <i_2,j_2> & \text{if } j_2>j_1\vee(i_1=2\wedge j_1=j_2) \\ <i_1,j_1> & \text{if } j_2=j_1\wedge i_1=i_2 \\ <2,i_1+1> & \text{otherwise} \end{cases}$$

(e) Suppose $SemiFmla\,(w)$, $Semiterm(v_2)$, $SemiFmla\,(v_1)$, $Quant(x)$ and $BVar(y)$ where $w=(0*\overline{LParen}*\overline{LParen}*x*y*\overline{LE})**(v_2*\overline{RParen})**(v_1*\overline{RParen})$. If $QCount(v_1)=0$ then $QCount(w)=0$. Otherwise define

$$QCount(w) = \begin{cases} QCount(v_1) & \text{if } LTerm\,(v_2)\vee\beta(1,QCount(v_1))=QType(x) \\ <QType(x),\beta(2,QCount(v_1))> & \text{if } \neg LTerm\,(v_2)\wedge\beta(1,QCount(v_1))=2 \\ <QType(x),1+\beta(2,QCount(v_1))> & \text{otherwise} \end{cases}$$

where $QType(x) = \begin{cases} 0 & \text{if } x=\overline{EQuant} \\ 1 & \text{otherwise} \end{cases}$

(f) If $w=(0*\overline{LParen}*\overline{LParen}*x*y*\overline{RParen})**(v_1\overline{RParen})$ where $Quant(x)$, $BVar(y)$, and $SemiFmla\,(v_1)$, then $QCount(w)=0$.

That completes the definition of *QCount*.

Another important operation we need to Σ_1^b–define in S_2^1 is the substitution of a term into a formula or term. First define

$$SubOK(w,x,v) \iff (FVar(x) \lor (BVar(x) \land Free(x,w))) \land \\ \land (SemiFmla(w) \lor Semiterm(w)) \land Semiterm(v) \land \\ \land (\forall i < Len(v))(\neg Bound(\beta(Si,v),w))$$

We define $Sub(w,x,v)$ to be the function satisfying:

$$z = Sub(w,x,v) \iff (\neg SubOK(w,x,v) \land z=0) \lor \\ \lor (UniqSeq(z) \land SubOK(w,x,v) \land \\ \land Len(z) = Len(w) + (Len(v) \dot{-} 1) \cdot ((\# i < Len(w))(\beta(Si,w)=x)) \land \\ \land (\forall j < Len(w))(\exists k < Len(z))[k = j + (Len(v) \dot{-} 1) \cdot ((\# i < j)(\beta(Si,w)=x)) \land \\ \land (\beta(Sj,w) \neq x \supset \beta(Sj,w) = \beta(Sk,v)) \land \\ \land (\beta(Sj,w) = x \supset (\forall u < Len(v))(\beta(u+k+1,z) = \beta(u+1,v)))])$$

so $Sub(w,x,v)$ is the result of substituting the term v for the variable x in w. We leave to the reader the proof that Sub is a Σ_1^b–defined function of S_2^1 (the existence and uniqueness conditions for the above defining equation must be proved in S_2^1.) We also claim that

$$S_2^1 \vdash Fmla(w) \land Term(v) \land SubOK(w,x,v) \supset \\ \supset Fmla(Sub(w,x,v)) \land QCount(w) = QCount(Sub(w,x,v)).$$

This is proved by a p-inductive proof.

In addition to the Sub function, we need a function for performing the simultaneous substitution of a vector of terms for a vector of variables. We define

$$VSubOK(w,x,v) \iff Seq(x) \land Seq(v) \land Len(x) = Len(v) \land \\ \land (\forall i < Len(x))[SubOK(w,\beta(i+1,x),\beta(i+1,v)) \land \\ \land (\forall j < Len(x))(j \neq i \supset \beta(j+1,x) \neq \beta(i+1,x)) \land \\ \land (\forall j < Len(v))(\forall k < Len(\beta(j+1,v)))(\beta(i+1,x) \neq \beta(k+1,\beta(j+1,v)))].$$

So $VSubOK(w,x,v)$ is true iff x is a vector of distinct variables, v is a vector of semiterms, no variable in x appears in any of the semiterms in v and if there are no bound variable conflicts which arise when the semiterms of v are substituted for the variables of x in w. We can now define $VSub$ by:

$$\begin{aligned} z{=}VSub(w,x,v) \iff {} & (\neg VSubOK(w,x,v)\wedge z{=}0)\vee \\ & \vee(VSubOK(w,x,v)\wedge Uniqseq(z)\wedge \\ & \quad\wedge(\exists y{\le}SqBd(w\#v,x))[Seq(y)\wedge Len(y){=}Len(x){+}1\wedge \\ & \qquad\wedge(\forall i{<}Len(x))(\beta(i{+}2,y){=}Sub(\beta(i{+}1,y),\beta(i{+}1,x),\beta(i{+}1,v)))\wedge \\ & \qquad\wedge\beta(1,y){=}w\wedge z{=}\beta(Len(x){+}1,y)]) \end{aligned}$$

We will omit proving the existence and uniqueness conditions for *VSub*, since the proof is straightforward with the machinery developed above and in Chapter 2.

We define cedents by the following p-inductive definition:

(a) $Cedent(0)$ (this is the empty cedent).

(b) If $Fmla(w)$ then $Cedent(w)$.

(c) If $Fmla(v_1)$, $Cedent(v_2)$ and $v_2{\neq}0$ then $Cedent((v_1{*}\overline{Separ}){*}{*}v_2)$.

(d) $Cedent(w)$ holds only as required by clauses (a)-(c).

Next we define a couple of functions for manipulating cedents:

$$CedentLen(w) = \begin{cases} 0 & \text{if } Len(w){=}0 \\ 1{+}(\#i{<}Len(w))Separ(\beta(Si,w)) & \text{otherwise} \end{cases}$$

$$\begin{aligned} Cedent\beta(a,w){=}v \iff {} & (v{=}CedentLen(w)\wedge(a{=}0\vee\neg Cedent(w)))\vee(a{\neq}0\wedge Cedent(w)\wedge \\ & \wedge(\exists x{\le}Len(w))(\exists y{\le}Len(w))[x{=}(\mu i{<}Len(w))(a{=}1{+}(\#j{<}i)Separ(\beta(Sj,w)))\wedge \\ & \quad\wedge y{=}(\mu i{<}Len(w))(a{=}(\#j{\le}i)Separ(\beta(Sj,w)))\wedge \\ & \quad\wedge v{=}Subseq(w,Sx,Sy)] \end{aligned}$$

So $Cedent\beta(a,w)$ is equal to the a-th formula of the cedent w, unless $a{=}0$ in which case it is equal to the number of formulae in w. Sequents are defined by

$$\begin{aligned} Sequent(w) \iff {} & (\exists u{<}Len(w))(Arrow(\beta(Su,w))\wedge Cedent(Subseq(w,1,Su))\wedge \\ & \wedge Cedent(Subseq(w,u{+}2,Len(w){+}1))) \end{aligned}$$

$$Arrowptr(w) = 1{+}(\mu i{<}Len(w))Arrow(\beta(i{+}1,w))$$

$$Antecedent(w) = \begin{cases} 0 & \text{if } \neg Sequent(w) \\ Subseq(w,1,Arrowptr(w)) & \text{otherwise} \end{cases}$$

$$Succedent(w) = \begin{cases} 0 & \text{if } \neg Sequent(w) \\ Subseq(w,1{+}Arrowptr(w),1{+}Len(w)) & \text{otherwise} \end{cases}$$

We define $QClass$ and $QBded$ as a function and predicate which count the number of alternations of quantifiers in sequents (and later in proofs). They are defined p-inductively by:

(a) If $Fmla(w)$ then

$$QClass(w) = \beta(2, QCount(w))$$

$$QBded(w) \iff QCount(w) \neq 0$$

(b) If $Cedent(w)$ and $w=(v_1 * \overline{Separ}) ** v_2$, then

$$QClass(w) = \max(QClass(v_1), QClass(v_2))$$

$$QBded(w) \iff QBded(v_1) \wedge QBded(v_2)$$

(c) If $Sequent(w)$ then

$$QClass(w) = \max(QClass(Antecedent(w)), QClass(Succedent(w)))$$

$$QBded(w) \iff QBded(Antecedent(w)) \wedge QBded(Succedent(w))$$

So $QBded(w)$ is true iff w includes no unbounded quantifiers. $QClass(w)$ is equal to the least i such that every formula in w is either a Σ_i^b– or a Π_i^b–formula.

We are now ready to metamathematically define what a proof is. A Gödel number of a proof codes a tree of sequents labeled precisely as to how the rules of inference are applied. Each node of the tree is labeled by an ordered pair $<x,w>$ where w is a formula and x codes the rule of inference used to deduce w from the sons of w (the sons of w are the sequents directly above w in the proof tree).

First, we define what the initial sequents of a proof may be. Let $LAxiom(v)$ be a predicate defined to be true iff $v=<0,w>$ where w is a logical axiom of one of the following forms:

(a) $A \longrightarrow A$ where A is an atomic formula.

(b) $\longrightarrow t=t$ where t is any term.

(c) $t=s \longrightarrow f(t)=f(s)$ where s and t are terms and f is one of the functions S, $\lfloor \frac{1}{2}x \rfloor$, or $|x|$.

(d) $t_1=s_1, t_2=s_2 \longrightarrow f(t_1,t_2)=f(s_1,s_2)$ where each s_i and t_i is a term and f is one of the functions $+$, $\cdot$, or $\#$.

(e) $t_1=s_1, t_2=s_2, p(t_1,t_2) \longrightarrow p(s_1,s_2)$ where each s_i and t_i is a term and p is one of the relations $\leq$ or $=$.

Let α be any unary Σ_1^b–definable function of S_2^1. We use α to enumerate a list of non-logical axioms and define $NLAxiom_\alpha(v)$ to be true iff (1) $v=<<v_0,v_1,v_2,v_3>,v_4>$, (2) either v_3 is the Gödel number of one of the finite number of *BASIC* axioms or $\alpha(v_2)=v_3$ and (3) the following four conditions hold: (a) $v_0=<x_1,\ldots,x_n>$ and $v_1=<y_1,\ldots,y_n>$, (b) for $1\le i\le n$, $BVar(x_i)$ and $Term(y_i)$, (c) $v_4=VSub(v_3,v_0,v_1)$, and (d) $Fmla(v_4)$. Thus the non-logical axioms are instances of formulae from *BASIC* axioms or formulae in the range of α. Note there is no conflict of variables in (c) since all variables in y_i are free variables and each x_i is a bound variable.

We are using α for additional generality; since every recursively enumerable set is the range of a polynomial time function, we can have any recursively enumerable set which includes the *BASIC* axioms as the set of axioms.

We now informally describe how proofs are arithmetized. A proof P is coded by a tree w. The root of w corresponds to the endsequent of P. The leaves of w correspond to the initial sequents of P. Each node of w corresponds to a sequent $\Gamma_n \longrightarrow \Delta_n$ of P. The sons of a node n of w correspond to the upper sequents of the inference in P which yielded $\Gamma_n \longrightarrow \Delta_n$. Accordingly, the valence of each node of w is not greater than two. The label on each node of w is $<x_n,v_n>$ where v_n is a Gödel number of the sequent $\Gamma_n \longrightarrow \Delta_n$ and x_n is a code detailing the inference used to derive that sequent. We already explained in detail what x_n is for initial sequents. For non-initial sequents, it suffices to take $x_n\le 23$ to be equal to the number of the inference as described in Chapter 4 or $x_n=24$ for a *PIND* inference.

To define proofs as metamathematical objects in S_2^1, we shall of course use a p-inductive definition. This is done by simultaneously defining the following predicates p-inductively.

$Proof_\alpha(w)$ $\Longleftrightarrow$ "w codes a proof with non-logical axioms specified by $NLAxiom_\alpha$ and all inductions in w are Δ_0^0-*PIND*'s."

$ProofFCF_\alpha(w)$ $\Longleftrightarrow$ "$Proof_\alpha(w)$ and there are no free cuts in w."

$QBded(w)$ $\Longleftrightarrow$ "All quantifiers in w are bounded."

$QClass(w)$ $\Longleftrightarrow$ "i is the least number such that all formulae in w are in $\Sigma_i^b \cup \Pi_i^b$."

$FreeForm(w,0,i)$ $\Longleftrightarrow$ "the i-th formula of the antecedent of the endsequent of w is a free formula."

$FreeForm(w,1,i)$ $\Longleftrightarrow$ "the i-th formula of the succedent of the endsequent of w is a free formula."

$i=INDType(w)$ $\Longleftrightarrow$ "$i\ge 1$ is the least number such that all induction inferences in w are Σ_{i-1}^b-*PIND* inferences, or $i=0$ and there are no induction inferences in w."

These can all be defined in a long but straightforward way by a p-inductive definition. Since it would not be very interesting to write out the definitions precisely, we omit them.

Some further useful predicates are:

$$
\begin{array}{lll}
ProofBQ^i & \iff & Proof_{\emptyset}(w) \wedge QClass(w) \leq i \wedge INDType(w) \leq i+1 \\
ProofBQ^i_\alpha & \iff & Proof_\alpha(w) \wedge QClass(w) \leq i \wedge INDType(w) \leq i+1 \\
ProofBD^i(w) & \iff & Proof_{\emptyset}(w) \wedge QBded(w) \wedge INDType(w) \leq i+1 \\
ProofBD^i_\alpha(w) & \iff & Proof_\alpha(w) \wedge QBded(w) \wedge INDType(w) \leq i+1 \\
ProofFCF^i(w) & \iff & ProofFCF_{\emptyset}(w) \wedge INDType(w) \leq i+1 \\
ProofFCF^i_\alpha(w) & \iff & ProofFCF_\alpha(w) \wedge INDType(w) \leq i+1
\end{array}
$$

When we use $\emptyset$ as a subscript, it denotes any function with range contained in the set of Gödel numbers of *BASIC* axioms. Thus $ProofBD^i(w)$, $ProofBQ^i(w)$ and $ProofFCF^i(w)$ each imply that w is a proof in the theory S^i_2. Also, $ProofBD^{(-1)}(w)$, $ProofBQ^{(-1)}(w)$ and $ProofFCF^{(-1)}(w)$ mean that w has no induction inferences at all. The difference between $ProofBD^i$ and $ProofBQ^i$ is that $ProofBD^i(w)$ means that w codes a bounded S^i_2–proof whereas $ProofBQ^i(w)$ means that w codes a bounded S^i_2–proof and that all the formulae in w are Σ^b_i– or Π^b_i–formulae.

Define the function $EndSequent\,(w)$ to be $\beta(2, Root\beta(w))$. Also define Prf as

$$
Prf(w,a) \iff a = EndSequent\,(w) \vee (0 * \overline{Arrow}) ** a = EndSequent\,(w).
$$

So $Prf(w,a)$ is true iff a is the Gödel number of the sequent or formula proved by the proof w. We further define:

$$
\begin{array}{lll}
Prf^i_\alpha(w,v) & \iff & Proof_\alpha(w) \wedge Prf(w,v) \wedge INDType(w) \leq i+1 \\
PrfFCF_\alpha(w,v) & \iff & ProofFCF_\alpha(w) \wedge Prf(w,v) \\
PrfFCF^i_\alpha(w,v) & \iff & ProofFCF^i_\alpha(w) \wedge Prf(w,v) \\
PrfFCF^i(w,v) & \iff & ProofFCF^i(w) \wedge Prf(w,v) \\
PrfBQ^i_\alpha(w,v) & \iff & ProofBQ^i_\alpha(w) \wedge Prf(w,v) \\
PrfBQ^i(w,v) & \iff & ProofBQ^i(w) \wedge Prf(w,v) \\
PrfBD_\alpha(w,v) & \iff & ProofBD_\alpha(w) \wedge Prf(w,v) \\
PrfBD^i_\alpha(w,v) & \iff & ProofBD^i_\alpha(w) \wedge Prf(w,v) \\
PrfBD^i(w,v) & \iff & ProofBD^i(w) \wedge Prf(w,v)
\end{array}
$$

$$
\begin{array}{lll}
Thm^i_\alpha(v) & \iff & (\exists w) Prf^i_\alpha(w,v) \\
ThmFCF_\alpha(v) & \iff & (\exists w) PrfFCF_\alpha(w,v) \\
ThmFCF^i_\alpha(v) & \iff & (\exists w) PrfFCF^i_\alpha(w,v) \\
ThmFCF^i(v) & \iff & (\exists w) PrfFCF^i(w,v) \\
ThmBQ^i_\alpha(v) & \iff & (\exists w) PrfBQ^i_\alpha(w,v) \\
ThmBQ^i(v) & \iff & (\exists w) PrfBQ^i(w,v) \\
ThmBD_\alpha(v) & \iff & (\exists w) PrfBD_\alpha(w,v) \\
ThmBD^i_\alpha(v) & \iff & (\exists w) PrfBD^i_\alpha(w,v) \\
ThmBD^i(v) & \iff & (\exists w) PrfBD^i(w,v)
\end{array}
$$

The last nine predicates are definitely not Δ^b_1 with respect to S^1_2 because of the unbounded quantifier $(\exists w)$. Hence they can not be used in principal formulae of induction inferences.

7.4. When Truth Implies Provability.

The main point of this section is to establish a crucial lemma for the Gödel incompleteness theorems.

Definition: $Num(x)$ is a function Σ_1^b–defined in S_2^1 so that $Num(x)$ is the Gödel number of the term I_x. We know that $Num(x)$ can be Σ_1^b–defined in S_2^1 since it is easy to give a p-inductive definition for Num.

From now on we will use $\ulcorner \chi \urcorner$ to denote the Gödel number of a term, formula, sequent, or proof χ. If $n \in \mathbf{N}$ then $\ulcorner n \urcorner$ denotes $\ulcorner I_n \urcorner$ or $Num(n)$.

We will write $FSub(\ulcorner A \urcorner, \ulcorner a \urcorner, t)$ to mean $Sub(\ulcorner A \urcorner, \ulcorner a \urcorner, Num(t))$; in other words, $FSub(\ulcorner A \urcorner, \ulcorner a \urcorner, t)$ is the formula obtained by replacing all occurrences of the free variable a in the formula A by the term I_t. If $\vec{a}$ is an n-tuple of free variables and $\vec{s}$ is an n-tuple of terms then we write

$$F\vec{S}ub(\ulcorner A \urcorner, \ulcorner \vec{a} \urcorner, \vec{s})$$

as an abbreviation for

$$FSub(\cdots(FSub(\ulcorner A \urcorner, \ulcorner a_1 \urcorner, s_1)\cdots), \ulcorner a_n \urcorner, s_n).$$

To improve readability, we shall frequently use $FSub$ implicitly in the following way. Let $A(a_1, \ldots, a_k)$ be a formula. Then $\ulcorner A(I_{s_1}, \ldots, I_{s_k}) \urcorner$ is an abbreviation for $F\vec{S}ub(\ulcorner A(\vec{a}) \urcorner, \ulcorner \vec{a} \urcorner, \vec{s})$. For example, we shall write

$$S_2^1 \vdash ThmBD^{(-1)}(\ulcorner A(I_{a_1}, \ldots, I_{a_k}) \urcorner)$$

as an abbreviation for

$$S_2^1 \vdash ThmBD^{(-1)}(F\vec{S}ub(\ulcorner A(a_1, \ldots, a_k) \urcorner, \ulcorner \vec{a} \urcorner, \vec{a})).$$

The next theorem is very important for establishing the Gödel incompleteness theorems.

Theorem 4:

(a) Let A be any Σ_1^b–formula in the language of Bounded Arithmetic. Let $a_1, \ldots, a_p$ be all the free variables of A. Then there is a term $t_A(\vec{a})$ such that

$$S_2^1 \vdash A(\vec{a}) \supset (\exists w \leq t_A) PrfFCF^{(-1)}(w, F\vec{S}ub(\ulcorner A \urcorner, \ulcorner \vec{a} \urcorner, \vec{a})).$$

(b) Let A be of the form $(\exists x)B(\vec{a},x)$ where B is a Σ_1^b-formula in the language of Bounded Arithmetic. Let $a_1, \ldots, a_p$ be all the free variables of A. Then

$$S_2^1 \vdash A(\vec{a}) \supset ThmFCF^{(-1)}(w, F\vec{S}ub(\ulcorner A \urcorner, \ulcorner \vec{a} \urcorner, \vec{a})).$$

So Theorem 4 asserts that for any Σ_1^b-formula $A(\vec{a})$, S_2^1 proves that for all values $\vec{n}$ such that $A(\vec{n})$ is true there is an induction free, free cut free proof of $A(I_{n_1}, \ldots, I_{n_k})$.

The proof of Theorem 4 is, of course, by induction on the complexity of A. The single hardest part to prove is Lemma 5:

Lemma 5: Let t be any term with free variables $a_1, \ldots, a_k$. Then

$$S_2^1 \vdash ThmFCF^{(-1)}(\ulcorner t(I_{a_1}, \ldots, I_{a_k}) = I_{t(a_1, \ldots, a_k)} \urcorner).$$

Proof: by induction on the complexity of t.

(a) Suppose t is the constant term 0. Then $S_2^1 \vdash ThmFCF^{(-1)}(\ulcorner 0=0 \urcorner)$ is immediate from the equality axioms.

(b) Suppose t is a variable symbol a. Then $S_2^1 \vdash ThmFCF^{(-1)}(\ulcorner I_a = I_a \urcorner)$ is immediate from the equality axioms.

(c) Suppose t is $S(r)$. By the induction hypothesis, S_2^1 proves that for all $\vec{n}$ there exists a proof that $r(I_{n_1}, \ldots, I_{n_k}) = I_{r(\vec{n})}$. So it suffices to show that

$$S_2^1 \vdash ThmFCF^{(-1)}(\ulcorner S(I_b) = I_{Sb} \urcorner).$$

This is proved by Σ_1^b-*PIND* with respect to b. Since there is a proof of $S(I_0)=I_1$, it is clearly true for $b=0$. To deal with the induction step, we argue informally inside S_2^1. The induction hypothesis is that there is a free cut free $S_2^{(-1)}$-proof of $S(I_{\lfloor \frac{1}{2}b \rfloor}) = I_{S(\lfloor \frac{1}{2}b \rfloor)}$. We divide the argument into two cases. First, suppose b is even. Then $S_2^{(-1)}$ proves immediately that $S(I_b) = I_b + S0$ and since $I_b + S0$ is identical to I_{Sb} this case is done. Second, suppose b is odd. Then $S_2^{(-1)}$ proves immediately that $S(I_b) = 2 \cdot I_{\lfloor \frac{1}{2}b \rfloor} + 2 = 2 \cdot (S(I_{\lfloor \frac{1}{2}b \rfloor}))$ and by combining that proof with the proof of $S(I_{\lfloor \frac{1}{2}b \rfloor}) = I_{S\lfloor \frac{1}{2}b \rfloor}$ we obtain, by an inessential cut, a proof of $S(I_b) = I_{Sb}$.

To apply Σ_1^b-*PIND* we must find a uniform bound t_S so that the proof of $S(I_b) = I_{Sb}$ is coded by a Gödel number $\le t_S(b)$. This is readily done, since in either case of the argument for the induction step, the difference in size of the proof of $S(I_b) = I_{Sb}$ over the size of the proof of $S(I_{\lfloor \frac{1}{2}b \rfloor}) = I_{S(\lfloor \frac{1}{2}x \rfloor)}$ is bounded by an amount proportional to the size $|b|$ of b. Thus the size of the free cut free $S_2^{(-1)}$-proof of $S(I_b) = I_{Sb}$ is quadratic in the size of b.

(d) Suppose t is $r+s$. As in (c), it will suffice to show that

$$S_2^1 \vdash ThmFCF^{(-1)}(\ulcorner I_b + I_c = I_{b+c} \urcorner).$$

Let b_u and c_u abbreviate $MSP(b,u)$ and $MSP(c,u)$ respectively. Let $D(u)$ be the Gödel number of the formula

$$I_{b_u} + I_{c_u} = I_{b_u + c_u}.$$

We will show that

$$S_2^1 \vdash ThmFCF^{(-1)}(D(\min(|b|,|c|)))$$

and

$$S_2^1 \vdash ThmFCF^{(-1)}(D(u)) \supset ThmFCF^{(-1)}(D(u \dot{-} 1)).$$

Then S_2^1 can use Σ_1^b-*LIND* to conclude $ThmFCF^{(-1)}(D(0))$, which is what we need to show, as $b_0 = b$ and $c_0 = c$.

We argue informally inside S_2^1. Let $v = \min(|c|,|b|)$; we want to show that $S_2^{(-1)}$ proves $D(v)$. Suppose without loss of generality that $v = |c|$. Then $c_v = 0$, so $D(0)$ is $\ulcorner I_{b_v} + 0 = I_{b_v + 0} \urcorner$, and this is easily proved in $S_2^{(-1)}$ by an equality axiom as I_{b_v} and I_{b_v+0} are the same term. We next argue the induction step. The induction hypothesis is that there is a free cut free $S_2^{(-1)}$-proof of $I_{b_u} + I_{c_u} = I_{b_u + c_u}$ and that $u > 0$. We want to show that there is an $S_2^{(-1)}$-proof of $I_{b_{u-1}} + I_{c_{u-1}} = I_{b_{u-1} + c_{u-1}}$. Note that $b_u = \lfloor \frac{1}{2} b_{u-1} \rfloor$ and $c_u = \lfloor \frac{1}{2} c_{u-1} \rfloor$. There are two cases to consider. First suppose that one of b_{u-1} and c_{u-1} is even and thus there is no carry from the rightmost bit position when they are added together. Then it is easy to add a small amount to the proof of $I_{b_u} + I_{c_u} = I_{b_u + c_u}$ to get a proof of $I_{b_{u-1}} + I_{c_{u-1}} = I_{b_{u-1} + c_{u-1}}$. Second, suppose that both b_{u-1} and c_{u-1} are odd. Then $S_2^{(-1)}$ can prove immediately from the *BASIC* axioms that $I_{b_{u-1}} + I_{c_{u-1}} = 2 \cdot (I_{b_u} + I_{c_u} + 1)$. We combine that with the $S_2^{(-1)}$-proof of $I_{b_u} + I_{c_u} = I_{b_u + c_u}$ using an inessential cut to get an $S_2^{(-1)}$-proof of $I_{b_{u-1}} + I_{c_{u-1}} = 2 \cdot (I_{b_u + c_u} + 1)$. By (c), there is an $S_2^{(-1)}$-proof of $I_{b_u + c_u} + 1 = I_{b_u + c_u + 1}$. From this we can use another inessential cut to obtain an $S_2^{(-1)}$-proof of $I_{b_{u-1}} + I_{c_{u-1}} = 2 \cdot I_{b_u + c_u + 1}$. Now we are done, since $2 \cdot I_{b_u + c_u + 1}$ and $I_{b_{u-1} + c_{u-1}}$ are the same term.

To apply Σ_1^b-*LIND* to conclude that $ThmFCF^{(-1)}(D(0))$ we must find a term t_+ which bounds the size of the $S_2^{(-1)}$-proofs constructed above. Because of the size bound established in (c), we know that the increase in size of the proof of $D(u \dot{-} 1)$ over the size of the proof of $D(u)$ is bounded by an amount quadratic in the size of $b_u + c_u$. Hence the size of the free cut free $S_2^{(-1)}$-proof of $I_b + I_c = I_{b+c}$ is bounded by a cubic polynomial of the sizes $|b|$ and $|c|$ of b and c.

(e) Suppose t is $r \cdot s$. As before, it suffices to show that S^1_2 proves

$$ThmFCF^{(-1)}(\ulcorner I_b \cdot I_c = I_{b \cdot c} \urcorner).$$

We shall prove this by using Σ^b_1–*PIND* with respect to the variable b.

We argue informally inside S^1_2. First we consider the case $b=0$; we want to show that $S_2^{(-1)}$ proves $I_0 \cdot I_c = I_0$. This is easily proved in $S_2^{(-1)}$ from the *BASIC* axioms, with a free cut free proof with size proportional to the size $|c|$ of c. We next do the induction step. The induction hypothesis is that there is an $S_2^{(-1)}$–proof of $I_{\lfloor \frac{1}{2}b \rfloor} \cdot I_c = I_{\lfloor \frac{1}{2}b \rfloor \cdot c}$ and we want to show that there is an $S_2^{(-1)}$ proof of $I_b \cdot I_c = I_{b \cdot c}$. There are two cases. First, if b is even then I_b is $I_{2 \cdot \lfloor \frac{1}{2}b \rfloor}$ and $I_{b \cdot c}$ is $I_{2 \cdot \lfloor \frac{1}{2}b \rfloor \cdot c}$. Hence the proof of $I_{\lfloor \frac{1}{2}b \rfloor} \cdot I_c = I_{\lfloor \frac{1}{2}b \rfloor \cdot c}$ is easily extended to a proof of $I_b \cdot I_c = I_{b \cdot c}$. Second, if b is odd then $2 \cdot \lfloor \frac{1}{2}b \rfloor + 1 = b$ and $S_2^{(-1)}$ can prove from the *BASIC* axioms that $I_b \cdot I_c = 2 \cdot I_{\lfloor \frac{1}{2}b \rfloor} \cdot I_c + I_c$. We combine this with the proof of $I_{\lfloor \frac{1}{2}b \rfloor} \cdot I_c = I_{\lfloor \frac{1}{2}b \rfloor \cdot c}$ using an inessential cut to get an $S_2^{(-1)}$–proof of $I_{2 \cdot \lfloor \frac{1}{2}b \rfloor \cdot c} + I_c = I_b \cdot I_c$. By (d), there is a free cut free $S_2^{(-1)}$–proof of $I_{2 \cdot \lfloor \frac{1}{2}b \rfloor \cdot c} + I_c = I_{b \cdot c}$ and we can use this and an inessential cut to get the desired $S_2^{(-1)}$–proof of $I_b \cdot I_c = I_{b \cdot c}$, which completes the induction step.

Since we used (d) in the induction step argument, the size of the free cut free $S_2^{(-1)}$–proof of $I_b \cdot I_c = I_{b \cdot c}$ constructed above is bounded by a quartic polynomial of the sizes $|b|$ and $|c|$ of b and c.

(f) Suppose t is $r\#s$. It suffices to show that S^1_2 proves

$$ThmFCF^{(-1)}(\ulcorner I_b \# I_c = I_{b\#c} \urcorner).$$

First, it is clear that if $b=0$ there is an $S_2^{(-1)}$–proof of this using the *BASIC* axioms. We shall prove the case $b>0$ in two parts. First, we show by Σ^b_1–*PIND* with respect to c that there is a free cut free $S_2^{(-1)}$–proof of $I_1 \# I_c = I_{1\#c}$ for all c; second, we use Σ^b_1–*PIND* with respect to b to prove that there is a free cut free $S_2^{(-1)}$–proof of $I_b \# I_c = I_{b\#c}$ for all b and c. We shall argue informally inside S^1_2.

First, it is clear that there is an $S_2^{(-1)}$–proof of $I_1 \# I_1 = I_2$ (since $1\#1=2$). So suppose there is a free cut free $S_2^{(-1)}$–proof of $I_1 \# I_{\lfloor \frac{1}{2}c \rfloor} = I_{1\#\lfloor \frac{1}{2}c \rfloor}$ where $c>1$. From the *BASIC* axioms, $S_2^{(-1)}$ proves $I_1 \# I_c = 2 \cdot I_{1\#\lfloor \frac{1}{2}c \rfloor}$, thus there is a free cut free $S_2^{(-1)}$–proof of $I_1 \# I_c = 2 \cdot I_{1\#\lfloor \frac{1}{2}c \rfloor}$. But $2 \cdot I_{1\#\lfloor \frac{1}{2}c \rfloor}$ is the same term as $I_{1\#c}$ and we are done.

Second, suppose there is a free cut free $S_2^{(-1)}$–proof of $I_{\lfloor \frac{1}{2}b \rfloor} \# I_c = I_{\lfloor \frac{1}{2}b \rfloor \# c}$ and $b \geq 2$. From the *BASIC* axioms, $S_2^{(-1)}$ proves $I_b \# I_c = (I_{\lfloor \frac{1}{2}b \rfloor} \# I_c) \cdot (1 \# I_c)$. Thus $S_2^{(-1)}$ proves $I_b \# I_c = I_{\lfloor \frac{1}{2}b \rfloor \# c} \cdot I_{1\#c}$. Hence, by (e), there is a free cut free $S_2^{(-1)}$–proof of $I_b \# I_c = I_{b\#c}$.

The size of the free cut free $S_2^{(-1)}$–proof constructed above is bounded by a fifth-order polynomial of the lengths $|b|$ and $|c|$ of b and c.

(g) Suppose t is $\lfloor \frac{1}{2}s \rfloor$ or t is $|s|$. It suffices to show that

$$ThmFCF^{(-1)}(\ulcorner I_{\lfloor \frac{1}{2}b \rfloor} = \lfloor \tfrac{1}{2} I_b \rfloor \urcorner)$$

and

$$ThmFCF^{(-1)}(\ulcorner I_{|b|} = |I_b| \urcorner).$$

These are easily proved by using Σ_1^b–$PIND$ with respect to b. We omit the details.

Q.E.D. □

We are now prepared to prove Theorem 4.

Proof: of Theorem 4 is by induction on the complexity of the formula A. We use separate cases depending on the outermost connective of A.

(a) Suppose A is an atomic formula or the negation of an atomic formula. A must be $t{=}s$, $\neg t{=}s$, $t \le s$, or $\neg t \le s$. By Lemma 5, S_2^1 proves that $t(I_{a_1}, \ldots, I_{a_k}) = I_{t(a_1,\ldots,a_k)}$ and $s(I_{a_1}, \ldots, I_{a_k}) = I_{s(a_1,\ldots,a_k)}$. So it will suffice to show that S_2^1 proves the following four formulae:

$$b{=}c \supset ThmFCF^{(-1)}(\ulcorner I_b{=}I_c \urcorner)$$

$$\neg b{=}c \supset ThmFCF^{(-1)}(\ulcorner \neg I_b{=}I_c \urcorner)$$

$$b \le c \supset ThmFCF^{(-1)}(\ulcorner I_b \le I_c \urcorner)$$

$$\neg b \le c \supset ThmFCF^{(-1)}(\ulcorner \neg I_b \le I_c \urcorner)$$

These are readily proved by induction on the lengths of b and c. The sizes of the free cut free $S_2^{(-1)}$–proofs are bounded by a quadratic polynomial of the lengths $|b|$ and $|c|$ of b and c.

(b) Suppose A is $B(\vec{a}) \vee C(\vec{a})$ and that Theorem 4 has already been established for B and C. Thus,

$$S_2^1 \vdash B(\vec{a}) \supset (\exists w \le t_B) PrfFCF^{(-1)}(w, F\vec{S}ub(\ulcorner B \urcorner, \ulcorner \vec{a} \urcorner, \vec{a}))$$

and

$$S_2^1 \vdash C(\vec{a}) \supset (\exists v \le t_C) PrfFCF^{(-1)}(v, F\vec{S}ub(\ulcorner C \urcorner, \ulcorner \vec{a} \urcorner, \vec{a})).$$

But it is easy for S_2^1 to prove that, given such a proof v or w, adding an ($\vee$:right) inference gives a $S_2^{(-1)}$–proof of $A(I_{a_1}, \ldots, I_{a_k})$. The bounding term t_A is easily obtained from t_B and t_C.

(c) Suppose A is $B \wedge C$. The argument for this case is similar to the argument for (b).

(d) Suppose A is $(\forall x \leq |s|)B(\vec{a},x)$. By the induction hypothesis, S_2^1 proves

$$B(\vec{a},b) \supset (\exists w \leq t_B(\vec{a},b)) PrfFCF^{(-1)}(w, F\vec{S}ub(\ulcorner B(\vec{a},b) \urcorner, \ulcorner \vec{a} \urcorner, \ulcorner b \urcorner, \vec{a}, b)).$$

We let $t(\vec{a}) = \sigma[t_B](\vec{a}, |\vec{s}|)$. Then by use of Σ_1^b-$LIND$ with respect to u, S_2^1 proves

$$A \wedge u \leq |s| \supset (\exists w \leq r(\vec{a},u)) PrfFCF^{(-1)}(w, F\vec{S}ub(\ulcorner b \leq u \longrightarrow B(\vec{a},b) \urcorner, \ulcorner \vec{a} \urcorner, \ulcorner u \urcorner, \vec{a}, u))$$

where $r(\vec{a},u) = 2^u \# (\gamma \cdot t)$, where γ is a suitable constant. This is because the proofs of $B(I_{a_1}, \ldots, I_{a_k}, I_b)$ for $b \leq u$ can be put together via inessential cut inferences to obtain a free cut free $S_2^{(-1)}$-proof of $b \leq I_u \longrightarrow B(I_{a_1}, \ldots, I_{a_k}, b)$.

By Lemma 5, S_2^1 proves

$$(\exists v \leq t_s(\vec{a})) PrfFCF^{(-1)}(v, \ulcorner I_{|s(\vec{a})|} = |s(I_{a_1}, \ldots, I_{a_k})| \urcorner).$$

By using a $(\forall\leq$:right) inference and another inessential cut, we can combine the proofs of $I_{|s|} = |s(I_{a_1}, \ldots, I_{a_k})|$ and of $b \leq I_{|s|} \longrightarrow B(I_{a_1}, \ldots, I_{a_k}, b)$ to get a free cut free $S_2^{(-1)}$-proof of $A(I_{a_1}, \ldots, I_{a_k})$. The bounding term t_A is easily obtained from r and t_s.

(e) Suppose A is $(\exists x \leq s)B(\vec{a},x)$. By the induction hypothesis, S_2^1 proves

$$B(\vec{a},b) \supset (\exists w \leq t_B) PrfFCF^{(-1)}(F\vec{S}ub(\ulcorner B \urcorner, \ulcorner \vec{a} \urcorner, \ulcorner b \urcorner, \vec{a}, b)).$$

Thus S_2^1 proves

$$A(\vec{a}) \supset (\exists x \leq s)(\exists w \leq t_B) PrfFCF^{(-1)}(F\vec{S}ub(\ulcorner B \urcorner, \ulcorner \vec{a} \urcorner, \ulcorner b \urcorner, \vec{a}, x)).$$

We argue informally in S_2^1. Suppose $A(\vec{a})$. Then we have just shown that there are an $x \leq s$ and a w so that w codes a free cut free $S_2^{(-1)}$-proof of $B(I_{a_1}, \ldots, I_{a_k}, I_x)$. By Case (a), there is a free cut free $S_2^{(-1)}$-proof of $I_x \leq s(I_{a_1}, \ldots, I_{a_k})$. We can combine these two proofs using an inessential cut and a $(\exists\leq$:right) inference to get a free cut free $S_2^{(-1)}$-proof of

$$(\exists x \leq s(I_{a_1}, \ldots, I_{a_k})) B(I_{a_1}, \ldots, I_{a_k}, x)$$

which is what we needed to show.

(f) Suppose A is $(\exists x)B(\vec{a},x)$. The proof for this case is similar to and slightly simpler than the proof for (e).

Q.E.D. □

It is important to recall that all formulae we are using in our arithmetization of metamathematics only use the original seven nonlogical symbols of Bounded Arithmetic; they do not contain any new Σ_1^b–defined functions or Δ_1^b–defined predicates. But of course any Δ_1^b–formula A which may include Σ_1^b–defined function symbols and Δ_1^b–defined predicate symbols is equivalent, provably in S_2^1, to two formulae A^Σ and A^Π where A^Σ and A^Π are in Σ_1^b and Π_1^b respectively and contain only the original seven nonlogical symbols of Bounded Arithmetic. This gives the following corollary to Theorem 4:

Corollary 6:
(a) Let A be any Δ_1^b–formula of S_2^1. That is, there are Σ_1^b–formulae A_1 and A_2 such that $S_2^1 \vdash A \leftrightarrow A_1$ and $S_2^1 \vdash A \leftrightarrow \neg A_2$. Then,

$$S_2^1 \vdash A(\vec{a}) \supset \mathit{ThmFCF}^{(-1)}(F\vec{S}ub(\ulcorner A_1 \urcorner, \ulcorner \vec{a} \urcorner, \vec{a}))$$

$$S_2^1 \vdash \neg A(\vec{a}) \supset \mathit{ThmFCF}^{(-1)}(F\vec{S}ub(\ulcorner \neg A_2 \urcorner, \ulcorner \vec{a} \urcorner, \vec{a})).$$

(b) Let A be any Δ_1^b–formula of S_2^1. Then

$$S_2^1 \vdash A(\vec{a}) \supset \mathit{ThmBD}^1(F\vec{S}ub(\ulcorner A \urcorner, \ulcorner \vec{a} \urcorner, \vec{a}))$$

$$S_2^1 \vdash \neg A(\vec{a}) \supset \mathit{ThmBD}^1(F\vec{S}ub(\ulcorner \neg A \urcorner, \ulcorner \vec{a} \urcorner, \vec{a})).$$

Note that in (b), ThmBD^1 is used instead of $\mathit{ThmFCF}^{(-1)}$. Unlike Theorem 4 and Lemma 5, Corollary 6(b) would still hold if we enlarged the syntax of our metamathematics to include symbols for Σ_1^b–defined functions and Δ_1^b–defined predicates.

Corollary 7: Let $A(b)$ be one of the formulae $\mathit{Thm}_\alpha(b)$, $\mathit{ThmBQ}_\alpha^i(b)$, $\mathit{ThmBD}^1(b)$, $\mathit{ThmBD}^i(b)$, etc. Then

$$S_2^1 \vdash (\forall x)[A(x) \supset \mathit{ThmBD}^1(FSub(\ulcorner A \urcorner, \ulcorner b \urcorner, x))].$$

7.5. Gödel Incompleteness Theorems.

Now that we have arithmetized the syntax of Bounded Arithmetic and, in particular, have proved Corollary 7, it will be straightforward to establish the Gödel incompleteness theorem. What we prove is somewhat stronger than the usual statements of the incompleteness results since we use *ThmFCF* instead of *Thm*; that is, we shall consider the consistency of free cut free proofs only, rather than of general proofs.

Lemma 8: (Gödel Diagonalization Lemma). Let $\psi(a)$ be any formula with one free variable a. Then there is a sentence ϕ such that

$$S_2^1 \vdash \phi \leftrightarrow \psi(\ulcorner \phi \urcorner).$$

Furthermore, if ψ is a Π_i^0-formula, then so is ϕ. If ψ is provably equivalent to a Σ_i^b-formula (resp. Π_i^b-formula) then so is ϕ.

Proof: Since Sub and Num are Σ_1^b-defined function symbols of S_2^1, Theorem 2.2 states that there is a formula $\chi(a)$ which is S_2^1-provably equivalent to $\psi(Sub(a,\ulcorner a \urcorner,Num(a)))$ such that if ψ is a Σ_i^b- (respectively, Π_i^b-) formula then so is χ. Define ϕ to be the sentence $\chi(I_{\ulcorner \chi \urcorner})$. So

$$S_2^1 \vdash \phi \leftrightarrow \psi(Sub(I_{\ulcorner \chi \urcorner},\ulcorner a \urcorner,Num(I_{\ulcorner \chi \urcorner}))).$$

By the definition of ϕ and the results of §7.4, we certainly have

$$S_2^1 \vdash I_{\ulcorner \phi \urcorner}=Sub(I_{\ulcorner \chi \urcorner},\ulcorner a \urcorner,Num(I_{\ulcorner \chi \urcorner}))$$

which shows that $S_2^1 \vdash \phi \leftrightarrow \psi(\ulcorner \phi \urcorner)$.

The fact that the quantifier structure of ϕ is the same as that of ψ is immediate from the fact that ϕ is a substitution instance of ψ and from Theorem 2.2. □

For added generality, we will work in theories stronger than S_2^i.

Definition: Let α be a unary Σ_1^b-defined function of S_2^1. We define $S_{2,\alpha}^i$ to be the theory such that

(a) The language of $S_{2,\alpha}^i$ is the language of Bounded Arithmetic.
(b) The axioms of $S_{2,\alpha}^i$ are the *BASIC* axioms plus all formulae with Gödel number in the range of α.
(c) $S_{2,\alpha}^i$ has all the Σ_i^b-*PIND* inference rules.

Example: Let PA be Peano arithmetic. Define β by

$$\beta(n) = \begin{cases} \ulcorner a=a \urcorner & \text{if } n \text{ is not a Gödel number of a } PA\text{-proof} \\ m & \text{if } n \text{ codes a } PA\text{-proof of the sequent with Gödel number } m \end{cases}$$

Then $S_{2,\beta}$ is equivalent to PA.

Definition: Let α be as above and fix $i \geq 1$. Since $PrfFCF_\alpha^i$ is Δ_1^b with respect to S_2^1, we can choose some formula $A \in \Pi_1^b$ such that $S_2^1 \vdash A(w,a) \leftrightarrow \neg PrfFCF_\alpha^i(w,a)$. Now let ψ be the formula $(\forall w)A(w,a)$. Define ϕ_α^i to be the formula whose existence is guaranteed by Lemma 8 such that

$$S_2^1 \vdash \phi_\alpha^i \leftrightarrow \psi(\ulcorner \phi_\alpha^i \urcorner).$$

Note that ϕ_α^i is a Π_1^0–formula of the form $(\forall w)B$ where B is a Π_1^b–formula which is Δ_1^b with respect to S_2^1. Also,

$$S_2^1 \vdash \phi_\alpha^i \leftrightarrow \neg ThmFCF_\alpha^i(\ulcorner \phi_\alpha^i \urcorner).$$

Theorem 9: (Gödel's First Incompleteness Theorem). Let α, $S_{2,\alpha}^i$ and ϕ_α^i be as above, with $i \geq 1$. Suppose $S_{2,\alpha}^i$ is consistent. Then,

$$S_{2,\alpha}^i \nvdash \phi_\alpha^i.$$

Proof: (by contradiction).

Suppose $S_{2,\alpha}^i \vdash \phi_\alpha^i$. Then by the cut elimination theorem (Theorem 4.3), there is a free cut free $S_{2,\alpha}^i$–proof of ϕ_α^i. Hence, by Corollary 7,

$$S_2^1 \vdash ThmFCF_\alpha^i(\ulcorner \phi_\alpha^i \urcorner).$$

From the assumption that $S_{2,\alpha}^i \vdash \phi_\alpha^i$ and the definition of ϕ_α^i,

$$S_{2,\alpha}^i \vdash \neg ThmFCF_\alpha^i(\ulcorner \phi_\alpha^i \urcorner),$$

and since $S_{2,\alpha}^i \supseteq S_2^1$, this contradicts the consistency of $S_{2,\alpha}^i$.

Q.E.D. □

Definition: The following predicates assert the consistency of various natural deduction proof systems:

$$\begin{array}{lll}
Con_\alpha^i & \Longleftrightarrow & \neg Thm_\alpha^i(\ulcorner \longrightarrow \urcorner) \\
ConBQ_\alpha^i & \Longleftrightarrow & \neg ThmBQ_\alpha^i(\ulcorner \longrightarrow \urcorner) \\
ConBQ^i & \Longleftrightarrow & \neg ThmBQ^i(\ulcorner \longrightarrow \urcorner) \\
ConBD_\alpha^i & \Longleftrightarrow & \neg ThmBD_\alpha^i(\ulcorner \longrightarrow \urcorner) \\
ConBD_\alpha & \Longleftrightarrow & \neg ThmBD_\alpha(\ulcorner \longrightarrow \urcorner)
\end{array}$$

$$
\begin{aligned}
&ConBD^i &&\Longleftrightarrow \neg ThmBD^i(\ulcorner \longrightarrow \urcorner)\\
&ConFCF^i_\alpha &&\Longleftrightarrow \neg(\exists v)[ThmFCF^i_\alpha(v) \wedge ThmFCF^i_\alpha((0*\overline{LParen}*\overline{Not})**(v*\overline{RParen}))]\\
&ConFCF_\alpha &&\Longleftrightarrow \neg(\exists v)[ThmFCF_\alpha(v) \wedge ThmFCF_\alpha((0*\overline{LParen}*\overline{Not})**(v*\overline{RParen}))]\\
&ConFCF^i &&\Longleftrightarrow \neg(\exists v)[ThmFCF^i(v) \wedge ThmFCF^i((0*\overline{LParen}*\overline{Not})**(v*\overline{RParen}))]
\end{aligned}
$$

For example, $ConFCF^1$ asserts that there is no formula A such that both A and $\neg A$ have free cut free S^1_2-proofs. It is necessary for our purposes that we define $ConFCF$ in this way; since Gentzen's cut elimination theorem can not be proved by Bounded Arithmetic, the fact that A and $\neg A$ have free cut free proofs does not *provably* imply that there is a free cut free proof of the empty sequent. Of course, a proof of the empty sequent is a proof of a contradiction, since the (Weak:right) inference may be used to infer anything from the empty sequent.

Definition: Let R be any axiomatizable theory of arithmetic. We write $Con(R)$, $BDCon(R)$, $BQCon(R)$ and $FCFCon(R)$ to denote formulae expressing various consistency properties of R. Thus, for example, we have:

$$
\begin{aligned}
&Con(S^i_2) &&\Longleftrightarrow Con^i\\
&Con(S^i_{2,\alpha}) &&\Longleftrightarrow Con^i_\alpha\\
&BQCon(S^i_2) &&\Longleftrightarrow ConBQ^i\\
&BDCon(S_{2,\alpha}) &&\Longleftrightarrow ConBD_\alpha\\
&BDCon(S^i_2) &&\Longleftrightarrow ConBD^i\\
&FCFCon(S_{2,\alpha}) &&\Longleftrightarrow ConFCF_\alpha\\
&FCFCon(S^i_2) &&\Longleftrightarrow ConFCF^i
\end{aligned}
$$

More generally, when R is any axiomatizable theory such that $R \supseteq S^1_2$, let α be a Σ^b_1-defined function of S^1_2 such that the range of α is equal to the set of Gödel numbers of theorems of R. Then $Con(R)$, $BDCon(R)$ and $FCFCon(R)$ are defined to be $Con(S^1_{2,\alpha})$, $BDCon(S^1_{2,\alpha})$ and $FCFCon(S^1_{2,\alpha})$, respectively.

In addition, the formulae Prf_R, Thm_R, $PrfBD_R$ and $PrfFCF_R$ will be used as alternative names for the formulae Prf^1_α, Thm^1_α, $PrfBD^1_\alpha$ and $PrfFCF^1_\alpha$, respectively.

Theorem 10: (Gödel's Second Incompleteness Theorem). Let α, ϕ^i_α and $S^i_{2,\alpha}$ be as above, with $i \geq 1$. Then,

$$S^1_2 \vdash \neg\phi^i_\alpha \supset \neg FCFCon(S^i_{2,\alpha})$$

and hence, if $S^i_{2,\alpha}$ is consistent,

$$S^i_{2,\alpha} \nvdash FCFCon(S^i_{2,\alpha}).$$

Proof: Because ϕ_α^i is a Π_1^0–formula of the form $(\forall w)A$ where A is a Π_1^b–formula which is Δ_1^b with respect to S_2^1, we have by Theorem 4 that

$$S_2^1 \vdash \neg\phi_\alpha^i \supset ThmFCF^{(-1)}(\ulcorner\neg\phi_\alpha^i\urcorner).$$

Also, by the definition of ϕ_α^i,

$$S_2^1 \vdash \neg\phi_\alpha^i \supset ThmFCF_\alpha^i(\ulcorner\phi_\alpha^i\urcorner).$$

It is also immediate from the definitions that

$$S_2^1 \vdash ThmFCF^{(-1)}(\ulcorner\neg\phi_\alpha^i\urcorner) \supset ThmFCF_\alpha^i(\ulcorner\neg\phi_\alpha^i\urcorner).$$

Putting these three formulae together, we get, from the definition of $FCFCon(S_{2,\alpha}^i)$, that

$$S_2^1 \vdash \neg\phi_\alpha^i \supset \neg FCFCon(S_{2,\alpha}^i).$$

Thus,

$$S_2^1 \vdash FCFCon(S_{2,\alpha}^i) \supset \phi_\alpha^i.$$

By the First Incompleteness Theorem, $S_{2,\alpha}^i \nvdash \phi_\alpha^i$, and hence

$$S_{2,\alpha}^i \nvdash FCFCon(S_{2,\alpha}^i).$$

Q.E.D. □

In Theorem 10 we only proved that $S_2^1 \vdash ConFCF_\alpha^i \supset \phi_\alpha^i$; we did not prove that $S_2^1 \vdash \phi_\alpha^i \supset ConFCF_\alpha^i$. In the standard treatments of Gödel's incompleteness theorem, Con_α is used instead of $ConFCF_\alpha$. Then if $\overline{\phi}_\alpha^i$ is defined using Thm_α^i in the same way that ϕ_α^i was defined from $ThmFCF_\alpha^i$, we have

$$S_2^1 \vdash \overline{\phi}_\alpha^i \leftrightarrow Con_\alpha^i$$

(see Theorem 5.6 of Feferman [9]). However, the author doubts that it is true that

$$S_2^1 \vdash \phi_\alpha^i \supset ConFCF_\alpha^i$$

since we are only considering free cut free proofs.

Since a free cut free proof with bounded initial sequents and bounded endsequent is a bounded proof, we have the following immediate corollary to the Second Incompleteness theorem.

Corollary 11: $(i \geq 1)$.

(a) $S_2^i \nvdash FCFCon(S_2^i)$

(b) $S_2^i \nvdash BDCon(S_2^i)$

(c) $S_{2,\alpha}^i \nvdash Con(S_{2,\alpha}^i)$

(d) If all axioms of $S_{2,\alpha}^i$ are bounded, then $S_{2,\alpha}^i \nvdash BQCon(S_{2,\alpha}^i)$

(e) If all axioms of $S_{2,\alpha}^i$ are bounded, then $S_{2,\alpha}^i \nvdash BDCon(S_{2,\alpha}^i)$

Corollary 12: (Gödel).

(a) Let PA be Peano arithmetic. Then $PA \nvdash Con(PA)$.

(b) If R is an axiomatizable theory which is stronger than S_2^1, then $R \nvdash Con(R)$.

Proof: First note (b) implies (a). Let α be a unary function Σ_1^b–definable in S_2^1 such that the range of α is equal to the set of Gödel numbers of theorems of R. Then $S_{2,\alpha}^1$ is equivalent to R. Thus (b) follows immediately from Corollary 11. □

7.6. Further Incompleteness Results.

In the author's opinion, the most important open question concerning Bounded Arithmetic is whether the hierarchy $S_2^1, S_2^2, \ldots$ of theories is proper. The results of this section were motivated by a desire to answer this question.

Let PA_k denote the subsystem for Peano arithmetic (PA) obtained by restricting induction to Σ_k^0– and Π_k^0–formulae. It is a classical result that $PA_{k+1} \vdash Con(PA_k)$. This can be proved by showing that PA_1 can formalize the proof of the cut elimination theorem and that PA_{k+1} can define a truth valuation on Σ_k^0– and Π_k^0–formulae. Consequently, $PA_{k+1} \vdash Con(PA_k)$. From this, it follows immediately that PA_{k+1} is strictly stronger that PA_k since by the Gödel incompleteness theorem, $PA_k \nvdash Con(PA_k)$.

One way we might prove that S_2^i is not equivalent to S_2^{i+1} would be to adapt the proof that PA_k is not equivalent to PA_{k+1}. Now it is certainly false that $S_2^{i+1} \vdash Con(S_2^i)$; indeed, $S_2 \nvdash Con(Q)$, where Q is Robinson's open, induction free subtheory of PA, (this is shown by Nelson [19] and Wilkie-Paris [31].) But instead, we might try to show that $S_2^{i+1} \vdash BDCon(S_2^i)$ or $S_2^{i+1} \vdash FCFCon(S_2^i)$. This would certainly suffice, since by Corollary 11, S_2^i does not prove either of these. However, as we show below, it is not true that for all $i \geq 1$, $S_2^{i+1} \vdash BDCon(S_2^i)$. The author does not know whether $S_2^{i+1} \vdash FCFCon(S_2^i)$, but he conjectures that it is not the case.

Definition: In order to improve readability, we shall use the symbols $\overset{\text{BD}}{\vdash}$ to denote "proves by bounded proof". This symbol will only be used metamathematically. For example, if Ψ is a bounded formula,

$$S^i_{2,\alpha} \overset{\text{BD}}{\vdash} \Psi$$

denotes the formula

$$(\exists w) PrfBD^i_\alpha(w, \ulcorner \Psi \urcorner)$$

which is a formula that asserts that there is a bounded $S^i_{2,\alpha}$–proof of Ψ.

If Ψ is not a bounded formula, we can still sometimes define a formula $S^i_{2,\alpha} \overset{\text{BD}}{\vdash} \Psi$. Namely, if Ψ is $(\forall x)A(x)$, let a be a new free variable. Then $S^i_{2,\alpha} \overset{\text{BD}}{\vdash} (\forall x)A(x)$ is defined to be the formula

$$S^i_{2,\alpha} \overset{\text{BD}}{\vdash} A(a)$$

where a is a new free variable not appearing in A. If Ψ is $(\exists y)A(y)$ and A is a bounded formula then $S^i_{2,\alpha} \overset{\text{BD}}{\vdash} \Psi$ is defined to be the formula

$$(\exists w)(\exists v)[Term(v) \wedge PrfBD^i_\alpha(w, Sub(\ulcorner (\exists y \le b)A \urcorner, \ulcorner b \urcorner, v))]$$

where b is a new free variable not appearing in A. In particular, we shall frequently have $\Psi = (\forall x)(\exists y)A(x,y)$ and in this case $S^i_{2,\alpha} \overset{\text{BD}}{\vdash} \Psi$ is the formula which asserts that there is a term t and a bounded $S^i_{2,\alpha}$–proof P such that P is a proof of $(\exists y \le t(a))A(a,y)$, where a is a new free variable.

Proposition 14: Let $\Psi(a)$ be any bounded formula. Suppose $S_2 \vdash (\forall x)\Psi(x)$. Then

$$S^1_2 \vdash (\forall x)[S^{(-1)}_2 \overset{\text{BD}}{\vdash} \Psi(I_x)].$$

By our conventions for abbreviating formulae, the conclusion of Proposition 14 is an abbreviation for

$$S^1_2 \vdash (\forall x)(\exists w) PrfBD^{(-1)}(w, FSub(\ulcorner \Psi \urcorner, \ulcorner a \urcorner, x)).$$

From now on, we shall use such abbreviation without comment and let the reader supply the translations.

Proof: This is proved by formalizing the proof of Theorem 4.10 inside S_2^1. We start with a bounded proof P with endsequent $\longrightarrow \Psi(a)$. By Theorem 4.9, P may be assumed to be restricted by parameter variables. S_2^1 can prove that, for any given value n for a, the induction inferences of P may be expanded to give an induction free proof of $\Psi(I_n)$.

One subtle point to notice is that this procedure is not provably uniform. That is, S_2^1 does not prove "Given a proof P of $(\forall x)\Psi(x)$ and given a number n, there is an induction free proof of $\Psi(I_n)$." Instead, given a proof P of $(\forall x)\Psi(x)$, S_2^1 proves "given a number n, there is an induction free proof of $\Psi(I_n)$."

Q.E.D. □

Definition: $S_2^1+BDCon(S_2^{(-1)})$ is the theory S_2^1 plus the bounded axiom $\neg PrfBD^{(-1)}(a,\ulcorner\longrightarrow\urcorner)$. Since $S_2^1+BDCon(S_2^{(-1)})$ is axiomatized by bounded formulae, it makes sense to discuss bounded proofs of that theory. We define

$$BDCon(S_2^1+BDCon(S_2^{(-1)}))$$

to be the formula expressing the bounded consistency of $S_2^1+BDCon(S_2^{(-1)})$.

Theorem 15: If $A(a_1,\ldots,a_k)$ is a Π_1^b–formula and if $S_2\vdash A$ then $S_2^1+BDCon(S_2^{(-1)})\vdash A$.

Proof: We assume without loss of generality that $k=1$. Since $A(a)$ is a Π_1^b–formula and $S_2\vdash A(a)$, Proposition 14 implies that

$$S_2^1\vdash(\forall x)[S_2^{(-1)}\overset{BD}{\models}A(I_x)].$$

On the other hand, by Theorem 4,

$$S_2^1\vdash[\neg A(a)\supset(S_2^{(-1)}\overset{BD}{\models}\neg A(I_a))].$$

Hence,

$$S_2^1+BDCon(S_2^{(-1)})\vdash(\forall x)A(x).$$

Q.E.D. □

Corollary 16: $S_2\nvdash BDCon(S_2^1+BDCon(S_2^{(-1)}))$.

Proof: By Gödel's Second Incompleteness Theorem, $S_2^1+BDCon(S_2^{(-1)})$ does not prove its own bounded consistency. Since $BDCon(S_2^1+BDCon(S_2^{(-1)}))$ is S_2^1–provably equivalent to a Π_1^b–formula, Corollary 16 is an immediate consequence of Theorem 15.

Q.E.D. □

Corollary 17: Either $S_2^1 \nvdash BDCon(S_2^{(-1)})$ or $S_2 \nvdash BDCon(S_2^1)$.

Proof: Suppose $S_2^1 \vdash BDCon(S_2^{(-1)})$. Then $S_2 \vdash [S_2^1 \models^{BD} BDCon(S_2^{(-1)})]$, and thus

$$S_2 \vdash BDCon(S_2^1) \leftrightarrow BDCon(S_2^1 + BDCon(S_2^{(-1)})).$$

So by Corollary 16, $S_2 \nvdash BDCon(S_2^1)$. □

Corollary 18: Let j be the least number (if any) such that $S_2^j \vdash BDCon(S_2^{(-1)})$. Then

(a) $S_2^i \nvdash BDCon(S_2^k)$, for all $i<j$ and all k.

(b) $S_2^i \nvdash BDCon(S_2^j)$, for all $i \geq j$.

Proof: (a) is obvious. (b) is proved in the same way as Corollary 17. □

Corollary 19: There is at most one $i>0$ such that $S_2^i \vdash BDCon(S_2^{i-1})$.

As we remarked at the outset of this section, these results were motivated by a desire to show that S_2^i and S_2^{i+1} are distinct theories. From this viewpoint, Corollary 19 is a negative result in that it states that the formula $BDCon(S_2^i)$ can not be used to separate the theories S_2^i and S_2^{i+1}.

There are weaker formulae we could attempt to use to separate S_2^i and S_2^{i+1}. For example, it is an open question whether S_2^{i+1} can prove $BQCon(S_2^i)$ or $FCFCon(S_2^i)$. The author conjectures that neither of $BQCon(S_2^{(-1)})$ and $FCFCon(S_2^{(-1)})$ is provable by S_2.

Chapter 8

A Proof-Theoretic Statement Equivalent to NP=co-NP

This chapter presents a reformulation of the NP=co–NP question in a proof-theoretic setting. It turns out that NP=co–NP is equivalent to the existence of a theory of Bounded Arithmetic satisfying a certain "anti-reflection" property.

Definition: Let $\phi(a,b,c)$ be the formula

$$(\forall y \le c)(\forall z \le a)\neg PrfBD^1(z, FSub(b, \ulcorner v \urcorner, y)).$$

Note that ϕ is a Π_1^b–formula, hence ϕ represents a co–NP predicate. It is not difficult to see that ϕ is co–NP complete.

Definition: Suppose NP=co–NP. Let ψ denote some fixed Σ_1^b–formula so that

$$\mathbf{N} \models \phi(a,b,c) \leftrightarrow \psi(a,b,c).$$

Definition: Suppose NP=co–NP. Let ϕ and ψ be as above. Then W is the theory with the same language as S_2^1 and all the axioms of S_2^1 plus the additional axioms:

(1) $\phi(a,b,c) \supset \psi(a,b,c)$
(2) $\psi(a,b,c) \supset \phi(a,b,c)$
(3) $\neg PrfBD^1(d, \ulcorner \longrightarrow \urcorner)$

Strictly speaking, W depends on the choice of ψ and a better notation for this theory might be W_ψ. However, we shall keep ψ fixed and suppress the subscript.

Definition: Let $\vec{a}$ be a vector $a_1, \ldots, a_n$. Then $I_{\vec{a}}$ denotes the vector $I_{a_1}, \ldots, I_{a_n}$.

The next proposition formalizes the claim that ϕ is co–NP complete.

Proposition 1: Suppose NP=co–NP. Let W be as above. Then

(a) W is a consistent extension of S_2^1.

(b) For every bounded formula $A(\vec{x})$, there is a Σ_1^b–formula A_Σ and a Π_1^b–formula A_Π such that

$$W \vdash [A(\vec{x}) \leftrightarrow A_\Sigma(\vec{x})] \wedge [A(\vec{x}) \leftrightarrow A_\Pi(\vec{x})].$$

Proof:

(a) Since all axioms of W are true (under the assumption that NP=co–NP) W must be consistent.

(b) Begin by supposing $A \in \Pi_1^b$. Since the Σ_1^b–replacement axioms are theorems of S_2^1 and by Corollary 2.15, there is a formula B which is Δ_1^b with respect to S_2^1 and a term $s(\vec{x})$ such that

$$S_2^1 \vdash A(\vec{x}) \leftrightarrow (\forall y \leq s(\vec{x}))B(\vec{x},y).$$

By Corollary 7.6(b), there are terms $r_1(\vec{a},b)$ and $r_2(\vec{a},b)$ such that

$$S_2^1 \vdash B(\vec{a},b) \supset (\exists z \leq r_1) PrfBD^1(z, \ulcorner B(I_{\vec{a}}, I_b) \urcorner)$$

and

$$S_2^1 \vdash \neg B(\vec{a},b) \supset (\exists z \leq r_2) PrfBD^1(z, \ulcorner \neg B(I_{\vec{a}}, I_b) \urcorner).$$

Since W has an axiom asserting $BDCon(S_2^1)$, we have

$$W \vdash B(\vec{a},b) \supset (\forall z) \neg PrfBD^1(z, \ulcorner \neg B(I_{\vec{a}}, I_b) \urcorner).$$

Let $t(\vec{a},b)$ be the term $\sigma[r_2]$. Then

$$W \vdash A(\vec{x}) \leftrightarrow (\forall y \leq s(\vec{x}))(\forall z \leq t(\vec{x},s(\vec{x}))) \neg PrfBD^1(z, \ulcorner \neg B(I_{\vec{x}}, I_y) \urcorner).$$

In other words,

$$W \vdash A(\vec{x}) \leftrightarrow \phi(t(\vec{x},s(\vec{x})), \ulcorner \neg B(I_{\vec{x}}, v) \urcorner, s(\vec{x})).$$

Let $C(\vec{x})$ be the formula $\psi(t(\vec{x},s(\vec{x})), \ulcorner \neg B(I_{\vec{x}}, v) \urcorner, s(\vec{x}))$. Then $C \in \Sigma_1^b$ and $W \vdash A \leftrightarrow C$. This establishes (b) for the case $A \in \Pi_1^b$.

If $A \in \Sigma_1^b$, apply the above construction to $\neg A$ to find $C \in \Sigma_1^b$ such that $W \vdash A \leftrightarrow \neg C$. So (b) holds for $A \in \Sigma_1^b$.

It is now easy to prove (b) for all bounded formulae A by induction on the quantifier complexity of A.

Q.E.D. □

Corollary 2: Suppose NP=co-NP and let W be as above. Then for every bounded formula A,

$$W \vdash [A(\vec{a}) \supset W \models^{BD} A(I_{\vec{a}})].$$

Proof: Let A_Σ be as in Proposition 1. Then

$$W \vdash [W \models^{BD} (A(\vec{a}) \leftrightarrow A_\Sigma(\vec{a}))].$$

Also, since $A_\Sigma \in \Sigma_1^b$ and by Theorem 7.4,

$$W \vdash [A_\Sigma(\vec{a}) \supset W \models^{BD} A_\Sigma(I_{\vec{a}})].$$

Hence,

$$W \vdash [A(\vec{a}) \supset W \models^{BD} A(I_{\vec{a}})].$$

Q.E.D. □

Proposition 3: Suppose R is a consistent theory extending S_2^1. Let $A(\vec{a})$ be any bounded formula in the language of S_2^1. If $R \vdash (\forall \vec{x}) A(\vec{x})$ then $\mathbf{N} \models (\forall \vec{x}) A(\vec{x})$.

Proof: Suppose $R \vdash (\forall \vec{x}) A(\vec{x})$ but $\mathbf{N} \models \neg A(\vec{n})$ for some fixed vector of integers $\vec{n}$. Then $S_2^1 \vdash \neg A(\vec{n})$ since $\neg A(\vec{n})$ is a closed, bounded, true formula. But since R is an extension of S_2^1, R must be inconsistent and we have arrived at a contradiction! □

Corollary 4: Suppose R is a consistent extension of S_2^1 and R is axiomatized by bounded formulae. Then every theorem of R is true for $\mathbf{N}$.

Definition: R is a *bounded* theory iff R is axiomatized by bounded formulae. The axioms of R may contain free variables.

So by Corollary 4, every bounded, consistent extension of S_2^1 has only true theorems.

Definition: Let R be a theory such that the language of R includes the language of Bounded Arithmetic. Then R is of ***polynomial growth rate*** iff whenever A is a bounded formula and $R \vdash (\forall \vec{x})(\exists y)A(\vec{x},y)$ there is a term $t(\vec{x})$ such that

$$R \vdash (\forall \vec{x})(\exists y \leq t(\vec{x}))A(\vec{x},y)$$

and such that t is a term in the language of Bounded Arithmetic.

Proposition 5: Let R be a bounded extension of S_2^1. Then R is of polynomial growth rate.

Proof: This is an immediate consequence of Parikh's theorem. □

We are now ready to state and prove the main theorem of this chapter.

Theorem 6: The following are equivalent:

(a) NP=co–NP.

(b) There is a bounded extension R of S_2^1 such that R is consistent and finitely axiomatized and such that for every bounded formula A,

$$R \vdash (\forall \vec{x})[A(\vec{x}) \supset R \vdash^{BD} A(I_{\vec{x}})].$$

(c) There is a consistent, axiomatizable extension R of S_2^1 which is of polynomial growth rate such that for every $A \in \Pi_1^b$,

$$R \vdash (\forall \vec{x})[A(\vec{x}) \supset R \vdash A(I_{\vec{x}})].$$

(d) There is a consistent extension R of S_2^1 such that for some polynomial $p(n_1,n_2,n_3)$,

$$\mathbf{N} \models [\phi(a,b,c) \supset (\exists z \leq 2^{p(|a|,|b|,|c|)}) Prf_R(z, \ulcorner \phi(I_a,I_b,I_c) \urcorner)].$$

Proof:

(a)$\Longrightarrow$(b): Let R be the theory W as in Corollary 2.

(b)$\Longrightarrow$(c): This is immediate from Proposition 5.

(c)$\Longrightarrow$(d): This is easily proved by noting that $\phi \in \Pi_1^b$, using the definition of polynomial growth rate and applying Proposition 3.

(d)$\Longrightarrow$(a): Suppose (d) holds. Since $\phi(a,b,c)$ is co–NP complete, it will suffice to show that $\phi(a,b,c)$ is in NP. By Proposition 3, if $R \vdash \phi(n_1,n_2,n_3)$ then $\mathbf{N} \models \phi(n_1,n_2,n_3)$. Hence

$$\mathbf{N} \models [\phi(a,b,c) \leftrightarrow (\exists z \leq 2^{p(|a|,|b|,|c|)}) Prf_R(z, \ulcorner \phi(I_a,I_b,I_c) \urcorner)].$$

The righthand side of this equivalence is a Σ_1^b formula and hence represents an NP predicate. Thus $\phi(a,b,c)$ is in NP.

Q.E.D. □

The importance of Theorem 6 is that it gives a reformulation of the NP=co–NP question in purely proof theoretic terms. The most striking equivalence is that of (a) and (b). The property expressed in (b) is a kind of "anti-reflection" property. So NP=co–NP is equivalent to the existence of a bounded theory with a certain "anti-reflection" property.

Trying to prove or disprove the statement (b) is a possible approach to resolving the NP vs. co–NP question. This approach does not suffer from the relativization results of Baker-Gill-Solovay [2] for the following reason: Consider a function f of polynomial growth rate such that NP^f=co–NP^f. If we have f as a new function symbol in R it may not be possible to axiomatize R so that there is a polynomial p such that

$$\mathbf{N} \models [f(a)=b \supset (\exists z \leq 2^{p(|a|,|b|)}) Prf_R(z, \ulcorner f(I_a)=I_b \urcorner)].$$

Theorem 6 inspires us to try some sort of self-referential formula $A(x)$ such that $A(x)$ is bounded and such that the theory R does not prove the existence of a proof or a disproof of $A(x)$. A natural choice for A is the formula $Con_R(x)$ which is defined as follows:

Definition: Let R be any axiomatizable theory. Then $Con_R(x)$ is defined to be the formula

$$(\forall y \leq x)(\neg Prf_R(y, \ulcorner \longrightarrow \urcorner)).$$

If R is furthermore a bounded theory, then $ConBD_R(x)$ is defined to be

$$(\forall y \leq x)(\neg PrfBD_R(y, \ulcorner \longrightarrow \urcorner)).$$

The question is whether there are "short" R-proofs of $Con_R(x)$ or $ConBD_R(x)$ for some bounded theory R. For example, if we could show that for all bounded, consistent, axiomatizable extensions R of S_2^1 there is no term $t(x)$ such that

$$\mathbf{N} \models (\forall x)(\exists y \leq t) PrfBD_R(y, \ulcorner Con_R(I_x) \urcorner)$$

then we would have shown that $NP \neq$co–NP. Unfortunately, we have the following result:

Proposition 7: Let R be any bounded, consistent, axiomatizable extension of S_2^1. Then there is a bounded, consistent, axiomatizable extension Q of R such that

$$Q \vdash (\forall x)[Q \overset{\text{BD}}{\vdash} Con_Q(I_x)].$$

Proposition 7 soundly destroys any hope of proving $NP \neq$co$-NP$ with the formula Con_Q since it is immediate that

$$Q \vdash (\forall x)[Con_Q(x) \supset (Q \overset{\text{BD}}{\vdash} Con_Q(I_x))].$$

Proof: Let $Q_0, Q_1, Q_2, \ldots$ be the following theories:

(a) Q_0 is R
(b) Q_1 is $Q_0 + Con(Q_0)$
(c) Q_{i+1} is $Q_i + Con(Q_i)$

Let Q be the theory $\bigcup_i Q_i$.

It is important to analyze exactly how $Q_0, Q_1, Q_2, \ldots$ are axiomatized. The theory Q_i is defined in a straightforward manner to have the axioms of R plus i additional axioms. Each axiom $Con(Q_i)$ is a formula with Gödel number G_i such that $2^{2^i} \leq G_i \leq 2^{2^{\delta \cdot i}}$ for each i and some constant δ. For each $i \geq 0$, S_2^1 can metamathematically discuss Q_i and S_2^1 can define formulae such as $Con(Q_i)$.

S_2^1 can also metamathetically define the theory Q in a straightforward manner. In particular, there is a Δ_1^b-predicate of S_2^1 which recognizes the axioms of Q.

Since each theory $Q, Q_0, Q_1, \ldots$ contains R, they each admit Σ_1^b-*PIND* inferences.

Now suppose we wish to find a Q-proof of $Con_Q(I_n)$ for some $n \in \mathbf{N}$. Let j_n be equal to the length of the length of n, i.e., $j_n = |(|n|)|$. Then for all $m > j_n$, the axiom $Con(Q_m)$ has Gödel number $G_m > n$. Hence, no axioms $Con(Q_m)$ where $m > j_n$ can appear in a Q-proof with Gödel number $\leq n$. Thus, a Q-proof with Gödel number $\leq n$ is in fact a Q_{j_n}-proof. S_2^1 can formalize this argument and hence

$$S_2^1 \vdash Con(Q_{j_n}) \supset Con_Q(n).$$

But now $Q \supseteq S_2^1$ and Q has $Con(Q_{j_n})$ as an axiom, so $Q \vdash Con_Q(n)$. The size of the S_2^1-proof of $Con(Q_{j_n}) \supset Con_Q(n)$ is proportional to the length $|n|$ of n and the size $|G_{j_n}|$ of the axiom $Con(Q_{j_n})$ is $\leq 2^{\delta \cdot j_n} \leq (1+|n|)^\delta$. Hence there is a polynomial, independent of n, such that the Gödel number of the Q-proof of $Con_Q(n)$ is less than $2^{p(|n|)}$.

Furthermore, S_2^1 and hence Q can formalize the reasoning of the above paragraph. Thus

$$S_2^1 \vdash (\forall x)[Q \overset{BD}{\vdash} Con_Q(I_x)].$$

So,

$$Q \vdash (\forall x)[Q \overset{BD}{\vdash} Con_Q(I_x)].$$

Q.E.D. □

Regarding Proposition 7 it should be noted (see Pudlak [23]) that

$$S_2^1 \vdash (\forall x)[S_2^1 \vdash Con_{S_2^1}(I_x)].$$

However, the author doubts that

$$S_2^1 \vdash (\forall x)[S_2^1 \overset{BD}{\vdash} Con_{S_2^1}(I_x)].$$

What Proposition 7 asserts is that for ***some*** bounded extension R of S_2^1,

$$R \vdash (\forall x)[R \overset{BD}{\vdash} Con_R(I_x)].$$

There are lower bounds known for the length of any R–proof of $Con_R(x)$. They were originally proved by H. Friedman [10] and later by Pudlak [23]. Their techniques can be extended to give a lower bound on the size of bounded R–proofs of $ConBD_R(x)$. Namely, we have:

Proposition 8: Let R be a bounded, consistent extension of S_2^1. Then for any term r there is a term q of the language of Bounded Arithmetic such that for all $n \in \mathbf{N}$ there is no bounded R–proof of $ConBD_R(q(I_n))$ with Gödel number less than $r(n)$.

Proof: by the method of H. Friedman [10] and Pudlak [23]. □

Unfortunately, the lower bound of Proposition 8 is not good enough to show that $NP \neq$co–NP and by Proposition 7 there is no way it can be improved significantly.

Proposition 7 destroyed our hope of using $A(x) = Con_R(x)$ to prove $NP \neq$co–NP. So what else can we try? Well, one possibility is to pick $A(x)$ to be some co–NP complete predicate. However, this is somewhat unsatisfactory; it would be preferable to find an $A(x)$ which is true for all x, since such a formula might be easier to manipulate.

Let PA and ZF denote the theories of Peano Arithmetic and Zermelo-Fraenkel set theory. H. Friedman has asked whether there are short PA-proofs of $Con_{ZF}(x)$. In an attempt to generalize his question, consider the following definition:

Definition: Let R be a consistent, bounded theory of arithmetic. Then the theory R', called the *jump of R* is defined so that

(1) The language of R' is the language of R plus a new predicate symbol T.

(2) All the axioms of R are axioms of R'.

(3) For every formula $A(\vec{a},\vec{b})$ in the language of R, the following is an axiom of R':

$$T(\ulcorner A(\vec{a},I_{\vec{b}})\urcorner) \leftrightarrow (\forall\vec{a})A(\vec{a},\vec{b}).$$

(4) In addition, R' has the axiom

$$Thm_R((0*\overline{Arrow})**a) \supset T(a).$$

It is clear that R' is an axiomatizable extension of R. The intended interpretation of the predicate $T(a)$ is "a is the Gödel number of a valid R-formula." As every axiom of R' is true for this interpretation, R' must be consistent.

We now consider the possibility of using $A(x)=Con_{R'}(x)$ to prove $NP{\neq}$co-NP. In this case we do not have the difficulties that arose in Proposition 7; namely, it is not the case that

$$R \vdash (\forall x)[R \vdash Con_{R'}(I_x)].$$

Indeed, it is not the case that

$$R' \vdash (\forall x)[R \vdash Con_{R'}(I_x)].$$

This is because $R' \vdash [(R \vdash Con_{R'}(I_a)) \supset Con_{R'}(a)]$ and by Gödel's second incompleteness theorem $R' \not\vdash (\forall x)Con_{R'}(x)$.

This inspires us to make the following conjecture:

Conjecture: For every bounded, consistent, axiomatized extension R of S_2^1,

$$R \not\vdash (\forall x)[Con_{R'}(x) \supset R \overset{\text{BD}}{\vdash} Con_{R'}(I_x)].$$

It should be difficult to prove this conjecture as an affirmative resolution of the conjecture would be a proof that $NP \neq \text{co}-NP$.

Chapter 9

Foundations of Second Order Bounded Arithmetic

Second order arithmetic is an extension of the first order theories discussed so far. In second order logic, we enlarge the formal system of logic to allow discussing functions and predicates directly. New second order variables refer to functions and predicates and allow quantification over functions and predicates.

Second order Bounded Arithmetic is different from the usual systems of second order arithmetic. There are restrictions on the functions used by second order Bounded Arithmetic; namely, the functions must have a polynomial growth rate. Also, the axioms of second order Bounded Arithmetic are much weaker that those of the usual second order theories of arithmetic. In particular, second order Bounded Arithmetic is **not** stronger than Peano arithmetic.

So why are we interested in such weak theories of Bounded Arithmetic? The classical second order theories have been motivated partially by a desire to develop mathematics on a logical basis more secure than set theory. Likewise, it is an interesting question how much of mathematics can be developed in second order Bounded Arithmetic; Nelson [19] and Hook [16] have worked on a closely related problem. However, we are interested in second order Bounded Arithmetic because we will establish results about the definability of functions which are analogous to our earlier theorems for first order Bounded Arithmetic. We shall define second order theories V_2^1 and U_2^1 such that a function f is $\Sigma_1^{1,b}$–definable in U_2^1 iff f is computable by a polynomial space bounded Turing machine (i.e., $f \in$PSPACE); similarly, f is $\Sigma_1^{1,b}$–definable in V_2^1 iff f is computable by an exponential time Turing machine (i.e., $f \in$EXPTIME).

This chapter defines the syntax and axioms of second order Bounded Arithmetic. We examine the question of using predicates versus functions as second order objects. Comprehension axioms and new induction axioms are introduced. Finally, the cut-elimination theorem is extended to second order theories of arithmetic. For cut-elimination, we must use natural deduction systems and accordingly we will define comprehension and induction rules as well as axioms.

In Chapter 10, the results relating second order Bounded Arithmetic to PSPACE and EXPTIME are obtained.

9.1. The Syntax of Second Order Bounded Arithmetic.

Although the reader should be somewhat familiar with second order logic, we shall review all the necessary syntax and terminology. For the most part, we follow the conventions of Chapter 3 of Takeuti [28].

The language of second order Bounded Arithmetic includes the first order language defined in Chapters 2 and 4. In addition, there are the following second order variables:

(1) Free and bound second order variables for predicates. For all $i,j \in \mathbf{N}$, α_i^j is a free j-ary second order predicate symbol and ϕ_i^j is a bound j-ary second order predicate symbol. We shall use $\alpha,\beta,\gamma,\ldots$ and $\phi,\chi,\psi,\ldots$ as metavariables for free and bound predicate variables, respectively.

(2) Free and bound second order variables for functions with polynomial growth rate. For every term t of the first order theory S_2 and for all $i,j \in \mathbf{N}$, $\varsigma_{i,j}^t$ is a free second order j-ary function variable and $\lambda_{i,j}^t$ is a bound second order j-ary function variable. We use $\varsigma^t,\eta^t,\theta^t,\ldots$ and $\lambda^t,\mu^t,\nu^t,\ldots$ as metavariables for free and bound second order function variables, respectively. These symbols range over functions f such that f is bounded by t; i.e., for all $\vec{x} \in \mathbf{N}^j$, $f(\vec{x}) \leq t(\vec{x})$.

Second order quantifiers are of the form $(\forall\phi)$, $(\exists\phi)$, $(\forall\lambda^t)$ and $(\exists\lambda^t)$. First order quantifiers are the same as before. The adjectives ***sharply bounded***, ***bounded*** and ***unbounded*** are used to describe first order quantifiers only. We shall occasionally not adhere precisely to the distinction between bound and free variables.

Definition: A *first order* formula is one with no second order quantifiers. Second order free variables may appear in a first order formula.

We classify second order formulae in a hierarchy of sets $\Sigma_i^{1,b}$, $\Pi_i^{1,b}$ of formulae:

Definition: A second order formula is *bounded* iff it contains no unbounded, first order quantifiers. The following sets of bounded second order formulae are defined inductively by:

(1) $\Sigma_0^{1,b} = \Pi_0^{1,b} = \Delta_0^{1,b}$ is the set of formulae which contain no second order quantifiers and no unbounded quantifiers (i.e., the set of bounded, first order formulae).

(2) $\Sigma_{i+1}^{1,b}$ is the set of formulae such that

(a) $\Pi_i^{1,b} \subseteq \Sigma_{i+1}^{1,b}$

(b) If A is in $\Sigma_{i+1}^{1,b}$, so are $(\forall x \leq t)A$, $(\exists x \leq t)A$, $(\exists\phi)A$ and $(\exists\lambda^t)A$.

(c) If A and B are in $\Sigma_{i+1}^{1,b}$, so are $A \wedge B$ and $A \vee B$.

(d) If $A \in \Sigma_{i+1}^{1,b}$ and $B \in \Pi_{i+1}^{1,b}$, then $\neg B$ and $B \supset A$ are in $\Sigma_{i+1}^{1,b}$.

(3) $\Pi_{i+1}^{1,b}$ is the set of formulae such that

(a) $\Sigma_i^{1,b} \subseteq \Pi_{i+1}^{1,b}$

(b) If A is in $\Pi_{i+1}^{1,b}$, so are $(\forall x \leq t)A$, $(\exists x \leq t)A$, $(\forall\phi)A$ and $(\forall\lambda^t)A$.

(c) If A and B are in $\Pi_{i+1}^{1,b}$, so are $A \wedge B$ and $A \vee B$.

(d) If $A \in \Pi_{i+1}^{1,b}$ and $B \in \Sigma_{i+1}^{1,b}$, then $\neg B$ and $B \supset A$ are in $\Pi_{i+1}^{1,b}$.

(4) $\Sigma_i^{1,b}$ and $\Pi_i^{1,b}$ are the smallest sets satisfying (1)-(3).

So $\Sigma_0^{1,b}$ is the set of bounded first order formulae which may contain second order free variables but no second order quantifiers. $\Sigma_i^{1,b}$ and $\Pi_i^{1,b}$ are defined by counting alternations of second order quantifiers ignoring first order bounded quantifiers.

It will be convenient to sometimes work in a theory which does not contain second order function variables. Accordingly, we define $\tilde{\Delta}_0^{1,b}$, $\tilde{\Sigma}_i^{1,b}$ and $\tilde{\Pi}_i^{1,b}$ to be the subsets of $\Delta_0^{1,b}$, $\Sigma_i^{1,b}$ and $\Pi_i^{1,b}$, respectively, containing just the formulae which contain no free or bound second order function variables.

In order to manipulate the second order variables and quantifiers in a natural deduction system we need additional inference rules:

(1) (second order $\forall$:left):

$$\frac{A(\alpha),\Gamma \longrightarrow \Delta}{(\forall\phi)A(\phi),\Gamma \longrightarrow \Delta}$$

and

$$\frac{A(\varsigma^i),\Gamma \longrightarrow \Delta}{(\forall\lambda^i)A(\lambda^i),\Gamma \longrightarrow \Delta}$$

(2) (second order $\forall$:right):

$$\frac{\Gamma \longrightarrow \Delta, A(\alpha)}{\Gamma \longrightarrow \Delta,(\forall\phi)A(\phi)}$$

and

$$\frac{\Gamma \longrightarrow \Delta, A(\varsigma^i)}{\Gamma \longrightarrow \Delta,(\forall\lambda^i)A(\lambda^i)}$$

where α and ς^i are the *eigenvariables* of the inferences and must not appear in the lower sequent.

(3) (second order $\exists$:left):

$$\frac{A(\alpha),\Gamma \longrightarrow \Delta}{(\exists\phi)A(\phi),\Gamma \longrightarrow \Delta}$$

and

$$\frac{A(\varsigma^t),\Gamma \longrightarrow \Delta}{(\exists\lambda^t)A(\lambda^t),\Gamma \longrightarrow \Delta}$$

where α and ς^t are the ***eigenvariables*** of the inferences and must not appear in the lower sequent.

(4) (second order $\exists$:right):

$$\frac{\Gamma \longrightarrow \Delta, A(\alpha)}{\Gamma \longrightarrow \Delta,(\exists\phi)A(\phi)}$$

and

$$\frac{\Gamma \longrightarrow \Delta, A(\varsigma^t)}{\Gamma \longrightarrow \Delta,(\exists\lambda^t)A(\lambda^t)}$$

Definition: Let A be a formula, $b_1, \ldots, b_m$ be free first order variables and $y_1, \ldots, y_m$ be bounded first order variables. Then $\{y_1, \ldots, y_m\}A(y_1, \ldots, y_m)$ is a meta-expression called the ***abstract*** of $A(b_1, \ldots, b_m)$. It is important to note that $\{\vec{y}\}A(\vec{y})$ is a meta-expression, so "{" and "}" are not symbols in the syntax of second order logic.

The idea of an abstract is that $\{\vec{y}\}A(\vec{y})$ specifies a predicate which is true for those $\vec{y}$ such that $A(\vec{y})$ holds. If $F(\alpha)$ is a formula containing the free second order predicate variable α, we use $F(\{\vec{y}\}A(\vec{y}))$ to denote the formula obtained by replacing every $\alpha(\vec{s})$ in F by $A(\vec{s})$. We will use metavariables $V, U, \ldots$ to denote abstracts. The formal definition of what $F(V)$ means is as follows:

Definition: If α is an n-ary predicate variable, $F(\alpha)$ is a formula and $V=\{y_1, \ldots, y_n\}A(y_1, \ldots, y_n)$ is an abstract, then $F(V)$ is the formula obtained by substituting V into F for α. $F(V)$ is defined by induction on the complexity of F:

(1) If α does not appear in F then $F(V)$ is F.

(2) If $F(\alpha)=\alpha(\vec{s})$, then $F(V)$ is $A(\vec{s})$.

(3) If F is $\neg B$, $B\wedge C$, $B\vee C$ or $B\supset C$. Then $F(V)$ is $\neg B(V)$, $B(V)\wedge C(V)$, $B(V)\vee C(V)$ or $B(V)\supset C(V)$ respectively.

(4) Suppose $F(\alpha)$ is $(\forall x)B(\alpha)$ or $(\exists x)B(\alpha)$. If x appears in A, we obtain A' by renaming the variable x in A to avoid conflict of variables. Then $F(V)$ is $(\forall x)B(\{\vec{y}\}A'(\vec{y}))$ or $(\exists x)B(\{\vec{y}\}A'(\vec{y}))$, respectively.

(5) Suppose $F(\alpha)$ is $(\forall\phi)B(\alpha)$ or $(\exists\phi)B(\alpha)$. If ϕ appears in A, we obtain A' by renaming

the variable ϕ in A to avoid conflict of variables. Then $F(V)$ is $(\forall\phi)B(\{\vec{y}\}A'(\vec{y}))$ or $(\exists\phi)B(\{\vec{y}\}A'(\vec{y}))$, respectively.

Proposition 1: Let F be a formula and U and V be abstracts. Any second order theory of arithmetic proves the sequent

$$(\forall x)(U(x)\leftrightarrow V(x)), F(U) \longrightarrow F(V).$$

Proof: This is Proposition 15.13 of Takeuti [28] and is easily proved by induction on the complexity of the formula F. □

Definition: Let $V=\{\vec{y}\}A(\vec{y})$ be an abstract. V is *atomic* iff A is atomic.

9.2. Comprehension Axioms and Rules.

The comprehension axiom of second order logic is fundamentally different from the axioms we used for first order Bounded Arithmetic. We define below comprehension rules as well as comprehension axioms.

Definition: Let Φ be a set of formulae. A Φ-abstract is one of the form $\{\vec{y}\}A(\vec{y})$ where A is in Φ. Φ is *closed under substitution* iff for every formula A in Φ and every Φ-abstract V, $A(V)$ is a formula in Φ.

We first define the comprehension axiom and rule for second order predicate symbols.

Definition: Let Φ be a set of formulae closed under substitution. The Φ *comprehension axioms*, Φ–CA, are given by the axiom scheme:

$$(\forall\vec{z})(\forall\vec{\phi})(\forall\vec{X}^{\bullet})(\exists\chi)(\forall\vec{y})\,[\chi(\vec{y})\leftrightarrow A(\vec{y},\vec{z},\vec{\phi},\vec{X}^{\bullet})]$$

where A must be in Φ.

Definition: Let Φ be a set of formulae closed under substitution. The Φ *comprehension rules*, Φ–CR, are inferences of the forms:

(1) (Φ–CR;$\exists$:right)

$$\frac{\Gamma \longrightarrow \Delta, F(V)}{\Gamma \longrightarrow \Delta, (\exists\phi)F(\phi)}$$

(2) (Φ–CR;$\forall$:left)

$$\frac{F(V),\Gamma \longrightarrow \Delta}{(\forall\phi)F(\phi),\Gamma \longrightarrow \Delta}$$

where in both (1) and (2), V must be a Φ-abstract. V is called the ***principal abstract*** of the inference.

Example: Let $F(\alpha)$ be the formula $(\exists y \leq a)(y \cdot y = a) \leftrightarrow \alpha(a)$. Then if A is $(\exists y \leq a)(y \cdot y = a)$, $F(A)$ is the formula

$$(\exists y \leq a)(y \cdot y = a) \leftrightarrow (\exists y \leq a)(y \cdot y = a).$$

Since $A \in \Sigma_0^{1,b}$, we can use $\Sigma_0^{1,b}$–CR to infer:

$$\frac{\longrightarrow (\forall x)[(\exists y \leq x)(y \cdot y = x) \leftrightarrow (\exists y \leq x)(y \cdot y = x)]}{\longrightarrow (\exists\phi)(\forall x)[(\exists y \leq x)(y \cdot y = x) \leftrightarrow \phi(x)]}$$

That is to say, $\Sigma_0^{1,b}$–comprehension implies that there is a predicate ϕ which is true precisely for the perfect squares.

Proposition 2: Let Φ be a set of formulae closed under substitution. Then the comprehension axioms Φ–CA are equivalent to the comprehension rules Φ–CR.

Proof: This is Theorem 15.16 of Takeuti [28]. One direction is easy. The example above provides the hint on how to prove the other direction, which is also easy. $\square$

We next define the comprehension axiom and rules for function symbols.

Definition: Let Φ be a set of formulae closed under substitution. The Φ ***function comprehension axioms***, Φ–FCA, are given by the following axiom scheme:

$$(\forall\vec{z})(\forall\vec{\phi})(\forall\vec{\eta}^{s})(\exists\lambda^{t})(\forall\vec{y})[A(\lambda^{t}(\vec{y}),\vec{y},\vec{z},\vec{\phi},\vec{\eta}^{s}) \leftrightarrow (\exists x \leq t)A(x,\vec{y},\vec{z},\vec{\phi},\vec{\eta}^{s})]$$

where A is any formula in Φ and t is any term.

Definition: Let Φ be a set of formulae closed under substitution. The Φ ***function comprehension rules***, Φ–FCR, are inferences of the form:

(1) (Φ–FCR;$\exists$:right)

$$\frac{\Gamma \longrightarrow \Delta, F(U)}{\Gamma \longrightarrow \Delta, (\exists \lambda^t) F(V)}$$

(2) Φ–FCR;$\forall$:left)

$$\frac{F(U), \Gamma \longrightarrow \Delta}{(\forall \lambda^t) F(V), \Gamma \longrightarrow \Delta}$$

where for both (1) and (2), t is any term, U must be an abstract of the form $\{\vec{y}\}(\exists x \leq t) A(x, \vec{y})$ and V must be the abstract $\{\vec{y}\} A(\lambda^t(\vec{y}), \vec{y})$, and A is required to be a formula in Φ. V is called the ***principal abstract*** of the inference.

Proposition 9: Let Φ be a set of formulae closed under substitution. The Φ–FCR rules are equivalent to the Φ–FCA axioms.

Proof:
$\Longrightarrow$. First we show that Φ–$FCR \Longrightarrow \Phi$–FCA. Let $A \in \Phi$. Using (Φ–FCR;$\exists$:right) we can infer

$$\begin{array}{c} \longrightarrow (\exists x \leq t) A(x, \vec{b}) \leftrightarrow (\exists x \leq t) A(x, \vec{b}) \\ \hline\hline \longrightarrow (\forall \vec{y})[(\exists x \leq t) A(x, \vec{y}) \leftrightarrow (\exists x \leq t) A(x, \vec{y})] \\ \hline \longrightarrow (\exists \lambda^t)(\forall \vec{y})[A(\lambda^t(\vec{y}), \vec{y}) \leftrightarrow (\exists x \leq t) A(x, \vec{y})] \end{array}$$

From this, the first and second order ($\forall$:right) inferences give the Φ–FCA axiom for A.
$\Longleftarrow$. The reverse implication is even easier.

Q.E.D. □

9.3. Axiomatizations of Second Order Bounded Arithmetic.

The weakest second order theories of Bounded Arithmetic are obtained by enlarging the first order theories S_2^i and T_2^i to include second order variables.

Definition: We define a hierarchy, $\Sigma_i^b(\alpha, \varsigma)$ and $\Pi_i^b(\alpha, \varsigma)$ of the second order formulae which contain no second order quantifiers. The definition of $\Sigma_i^b(\alpha, \varsigma)$ and $\Pi_i^b(\alpha, \varsigma)$ is completely analogous to the definition of Σ_i^b and Π_i^b in §2.1, the only difference being that free second order variables may appear without restriction in the formulae. The sets $\Sigma_i^b(\alpha)$ and $\Pi_i^b(\alpha)$ contain those formulae of $\Sigma_i^b(\alpha, \varsigma)$ and $\Pi_i^b(\alpha, \varsigma)$, respectively, which have no second order function variables.

Definition: $S_2^i(\alpha,\varsigma)$ is the second order theory with second order function and predicate variables and the following axioms:

(1) *BASIC* axioms,
(2) For each function variable ς_i^t, the axiom $(\forall\vec{x})(\varsigma_i^t(\vec{x})\leq t(\vec{x}))$,
(3) The $\Sigma_i^b(\alpha,\varsigma)$–*PIND* axioms.

$S_2^i(\alpha)$ is the second order theory with only second order predicate variables (but no second order function variables). The axioms for $S_2^i(\alpha)$ are:

(1) *BASIC* axioms,
(2) The $\Sigma_i^b(\alpha)$–*PIND* axioms.

$S_2(\alpha)$ is the union of the theories $S_2^i(\alpha)$ and $S_2(\alpha,\varsigma)$ is the union of the theories $S_2^i(\alpha,\varsigma)$.

$T_2^i(\alpha,\varsigma)$, $T_2^i(\alpha)$, $T_2(\alpha,\varsigma)$ and $T_2(\alpha)$ are defined similarly using the *IND* axioms instead of the *PIND* axioms.

All of our earlier work on S_2^i can be relativized to $S_2^i(\alpha,\varsigma)$. For example, the relativization of Theorem 2.6 holds and, for all $i\geq 1$, $S_2^i(\alpha,\varsigma)$ proves the $\Sigma_i^b(\alpha,\varsigma)$–*LIND* axioms. Another result which carries over is Theorem 2.7: by essentially the same proof as before we can show that $S_2^1(\alpha)$ can $\Sigma_1^b(\alpha)$–define the function

$$f(a) \;=\; (\#z\leq|a|)(\alpha(z)).$$

Also $S_2^i(\alpha,\varsigma)$ is an extension of the theories we used to discuss the relativized polynomial hierarchy in §5.4. In fact, it is now clear the function symbols $\eta_{j,k}^p$ of §5.4 were syntactically equivalent to second order function variables. Thus the theories $S_2^i(\alpha)$ and $S_2^i(\alpha,\varsigma)$ satisfy a relativized version of the Main Theorem 5.6.

Definition: U_2^i is the second order theory of Bounded Arithmetic which has second order predicate variables and function variables and which has the following axioms:

(1) All axioms of $S_2(\alpha,\varsigma)$,
(2) $\Sigma_0^{1,b}$–comprehension axioms, ($\Sigma_0^{1,b}$–CA and $\Sigma_0^{1,b}$–FCA),
(3) $\Sigma_i^{1,b}$–*PIND* axioms.

U_2 is the theory $\bigcup_i U_2^i$.

Definition: $\widetilde{U}_2^i$ is a second order theory of Bounded Arithmetic with predicate variables but no function variables. The axioms of $\widetilde{U}_2^i$ are

(1) All axioms of $S_2(\alpha)$,
(2) $\tilde{\Sigma}_0^{1,b}$–comprehension axioms ($\tilde{\Sigma}_0^{1,b}$–CA),
(3) $\tilde{\Sigma}_i^{1,b}$–$PIND$ axioms.

$\tilde{U}_2$ is the theory $\bigcup_i \tilde{U}_2^i$.

Definition: V_2^i, $\tilde{V}_2^i$, V_2 and $\tilde{V}_2$ are defined exactly like U_2^i, $\tilde{U}_2^i$, U_2 and $\tilde{U}_2$ (respectively) except that IND axioms are used instead of $PIND$ axioms.

Proposition 4: $(i \geq 1)$. $V_2^i \vdash U_2^i$ and $\tilde{V}_2^i \vdash \tilde{U}_2^i$.

Proof: $\Sigma_i^{1,b}$–$IND \Longrightarrow \Sigma_i^{1,b}$–$LIND$ is trivial. $\Sigma_i^{1,b}$–$LIND \Longrightarrow \Sigma_i^{1,b}$–$PIND$ is readily established by using the method of the proof of Theorem 2.11. These implications show that $V_2^i \vdash U_2^i$. $\tilde{V}_2^i \vdash \tilde{U}_2^i$ is proved by the same argument. □

The next theorem states that we can dispense with second order function variables if desired and just work with second order predicate variables.

Theorem 5: U_2^i is a conservative extension of $\tilde{U}_2^i$. V_2^i is a conservative extension of $\tilde{V}_2^i$.

Theorem 5 is proved by a series of lemmas. The most important one is Lemma 6:

Lemma 6: Let A be a $\Sigma_i^{1,b}$–formula with no free second order function variables. Then there is a $\tilde{\Sigma}_i^{1,b}$–formula A^* such that

$$U_2^0 \vdash A \leftrightarrow A^*.$$

Proof: The idea is that function variables in A can be replaced by predicate variables which encode the value of the function variables. We define a metaformula G such that $(\forall \vec{y})G(\varsigma^t,\alpha)$ asserts that the predicate α encodes the values of the function ς^t. When ς^t is k-ary, α must be $(k+1)$-ary and we define $G(\varsigma^t,\alpha)$ to be the formula

$$(\forall x < |t(\vec{y})|)(\alpha(x,\vec{y}) \leftrightarrow Bit(x,\varsigma^t(\vec{y}))=1).$$

So $(\forall \vec{y})G(\varsigma^t,\alpha)$ says that for all $x < |t(\vec{y})|$, $\alpha(x,\vec{y})$ is true iff the x-th bit of the binary expression for $\varsigma^t(\vec{y})$ is 1. Since $\varsigma^t(\vec{y}) \leq t(\vec{y})$ for all $\vec{y}$, α does indeed code the values of ς^t. G is a metaformula rather than a formula since the definition of $G(\varsigma^t,\alpha)$ depends on the term t and on the arity of ς^t.

Note that $G(\varsigma^t,\alpha)$ is a $\Sigma_0^{1,b}$–formula (in fact, $G(\varsigma^t,\alpha)$ is a $\Delta_1^b(\alpha,\varsigma)$–formula.) The $\Sigma_0^{1,b}$–CA axioms prove

$$(\forall\lambda^t)(\exists\phi)(\forall x)(\forall\vec{y})(\phi(x,\vec{y})\leftrightarrow Bit(x,\lambda^t(\vec{y}))=1),$$

hence, $U_2^0\vdash(\forall\lambda^t)(\exists\phi)(\forall\vec{y})G(\lambda^t,\phi)$.

Conversely, since $S_2^1(\alpha,\varsigma)\subseteq U_2^0$, we can introduce a new $\Sigma_1^b(\alpha,\varsigma)$–defined function symbol f_ϕ^t in U_2^0 satisfying

$$(\forall\vec{y})\,[|f_\phi^t(\vec{y})|\leq|t(\vec{y})|\wedge(\forall x<|t(\vec{y})|)(Bit(x,f_\phi^t(\vec{y}))=1\leftrightarrow\phi(x,\vec{y}))].$$

By the $\Sigma_0^{1,b}$–FCA axioms, U_2^0 can prove

$$(\forall\phi)(\exists\lambda^t)(\forall\vec{y})\,[(\exists z\leq t(\vec{y}))(z=\min(t(\vec{y}),f_\phi^t(\vec{y})))\leftrightarrow\lambda^t(\vec{y})=\min(t(\vec{y}),f_\phi^t(\vec{y}))].$$

Let $H(\varsigma^t,\alpha)$ be the metaformula $\varsigma^t(\vec{y})=\min(t(\vec{y}),f_\alpha^t(\vec{y}))$. It is now immediate that

$$U_2^0\vdash(\forall\phi)(\exists\lambda^t)(\forall\vec{y})H(\lambda^t,\phi)$$

and since $U_2^0\vdash G(\varsigma^t,\alpha)\supset H(\varsigma^t,\alpha)$,

$$U_2^0\vdash(\forall\lambda^t)(\exists\phi)H(\lambda^t,\phi).$$

We are now ready to construct the desired formula A^* equivalent to A. For every second order function variable $\lambda_j^{t_i}$ in A we use a new second order predicate variable $\psi_{i,j}$. We replace each $(\forall\lambda_j^{t_i})$ or $(\exists\lambda_j^{t_i})$ by $(\forall\psi_{i,j})$ or $(\exists\psi_{i,j})$, respectively. Let $h_{i,j}$ be the $\Sigma_1^b(\alpha)$–defined function such that

$$h_{i,j}(\vec{y}) \;=\; \min(t_i(\vec{s}),f_{\psi_{i,j}}^{t_i}(\vec{s})).$$

Wherever $\lambda_j^{t_i}(\vec{s})$ appears in A we replace it by $h_{i,j}(\vec{s})$. After all these replacements have been carried out we have the formula A^*. The $\Sigma_1^b(\alpha)$–defined function symbols $h_{i,j}$ can be removed by replacing them by their defining formulae.

It is clear that A^* is a $\tilde{\Sigma}_i^{1,b}$–formula and that $U_2^0\vdash A\leftrightarrow A^*$.

Q.E.D. □

Definition: $\hat{U}_2^i$ is the theory $\tilde{U}_2^i$ extended to include second order function variables and $\Sigma_0^{1,b}$–comprehension. (However, $\hat{U}_2^i$ does **not** include all the $\Sigma_i^{1,b}$–$PIND$ axioms.)

$\hat{V}_2^i$ is the theory $\tilde{V}_2^i$ extended to include second order function variables and $\Sigma_0^{1,b}$–comprehension (but not all the $\Sigma_i^{1,b}$–IND axioms.)

Lemma 7:

(a) $\hat{U}_2^i$ is a conservative extension of $\tilde{U}_2^i$.

(b) $\hat{V}_2^i$ is a conservative extension of $\tilde{V}_2^i$.

Proof: By the proof of Lemma 6, for every formula A there is a formula A^* such that $U_2^0 \vdash A \leftrightarrow A^*$ (even if A is not bounded). Furthermore, if $A \in \Sigma_i^{1,b}$ then $A^* \in \tilde{\Sigma}_i^{1,b}$. We claim that for all formulae A, if $\hat{U}_2^i \vdash A$ then $\tilde{U}_2^i \vdash A^*$. This will suffice to prove Lemma 7 as A^* is equal to A when A contains no second order function variables.

The claim is proved by induction on the number of inferences in a $\hat{U}_2^i$-proof of A. The only nontrivial case to consider is the $\Sigma_i^{1,b}$-*FCR* comprehension rules. However, $\tilde{U}_2^i$ can emulate $\Sigma_i^{1,b}$-*FCR* by using the $\Sigma_i^{1,b}$-*CR* comprehension rule. We leave the details to the reader. □

Lemma 8:

(a) The $\Sigma_i^{1,b}$-*PIND* axioms are theorems of $\hat{U}_2^i$.

(b) The $\Sigma_i^{1,b}$-*IND* axioms are theorems of $\hat{V}_2^i$.

Proof:

(a) This is immediate from Lemma 6 and the fact that $\hat{U}_2^i$ has the $\tilde{\Sigma}_i^{1,b}$-*PIND* axioms.

(b) is proved by the same argument. □

Proof: of Theorem 5:

By Lemma 8, $\hat{U}_2^i \equiv U_2^i$ and $\hat{V}_2^i \equiv V_2^i$. Hence, by Lemma 7, U_2^i is a conservative extension of $\tilde{U}_2^i$ and V_2^i is a conservative extension of $\tilde{V}_2^i$.

Q.E.D. □

In addition to the theories defined above, there are two more theories, $\mathring{U}_2^i$ and $\mathring{V}_2^i$, which are in some respects more natural choices for second order Bounded Arithmetic.

Definition: Let R be a second order theory of Bounded Arithmetic and let A be any formula. Then A is $\Delta_i^{1,b}$ ***with respect to the theory*** R iff there are formulae $B \in \Sigma_i^{1,b}$ and $C \in \Pi_i^{1,b}$ such that $R \vdash A \leftrightarrow B$ and $R \vdash A \leftrightarrow C$.

When it is clear what theory is being discussed we shall merely say A is $\Delta_i^{1,b}$ to mean that A is $\Delta_i^{1,b}$ with respect to R.

Definition: $\mathring{U}_2^i$ is a second order theory of Bounded Arithmetic with second order predicate variables but no function variables. The axioms of $\mathring{U}_2^i$ are

(1) All axioms of $\tilde{U}_2^i$

(2) $\tilde{\Delta}_1^{1,b}$-comprehension axioms ($\tilde{\Delta}_1^{1,b}$-*CA*).

$\mathring{U}_2$ is the theory $\bigcup_i \mathring{U}_2^i$.

Definition: $\mathring{V}_2^i$ and $\mathring{V}_2$ are defined analogously to $\mathring{U}_2^i$ and $\mathring{U}_2$. So $\mathring{V}_2^i$ and $\mathring{V}_2$ are the theories $\tilde{V}_2^i$ and $\tilde{V}_2$ (respectively) plus the $\tilde{\Delta}_1^{1,b}$–CA axioms.

It is an immediate consequence of Lemma 6 that second order function variables may be added to the syntax of $\mathring{U}_2^i$ or $\mathring{V}_2^i$ to obtain a conservative extension. Of course when we add second order function variables we may also use the $\Delta_1^{1,b}$–CA axioms and the $\Sigma_i^{1,b}$–$PIND$ or the $\Sigma_i^{1,b}$–IND (respectively) axioms. However, for our purposes in §10.5 and §10.6, it is more convenient to work with the theories $\mathring{U}_2^i$ and $\mathring{V}_2^i$ without second order function variables.

9.4. The Cut Elimination Theorem for Second Order Logic.

We next prove that Gentzen's cut elimination theorem holds for $\tilde{U}_2$ and $\tilde{V}_2$. We will show in §9.7 that $\mathring{U}_2$ and $\mathring{V}_2$ also satisfy a version of Gentzen's cut elimination theorem.

Definition: Let $A(a_1, \ldots, a_k, \alpha_1, \ldots, \alpha_n)$ be a formula with all free variables as indicated. We say that B is a ***substitution instance*** of A iff B is $A(t_1, \ldots, t_k, V_1, \ldots, V_n)$ where each t_i is an arbitrary term and each V_i is a $\Sigma_0^{1,b}$–abstract.

Lemma 9: ($i \geq 0$.)

(a) If A is a $\Sigma_i^{1,b}$– ($\Pi_i^{1,b}$–) formula then every substitution instance of A is a $\Sigma_i^{1,b}$– (respectively, $\Pi_i^{1,b}$–) formula.

(b) Suppose P is a $\tilde{U}_2^i$–proof (respectively, a $\tilde{V}_2^i$–proof) of $\Gamma \longrightarrow \Delta$ and that every principal formula of a free cut inference in P is a first order formula. Then there is a free cut free $\tilde{U}_2^i$–proof (respectively, $\tilde{V}_2^i$–proof) P^* of $\Gamma \longrightarrow \Delta$.

(c) Suppose P is a free cut free $\tilde{U}_2^i$–proof (respectively, $\tilde{V}_2^i$–proof) of $\Gamma \longrightarrow \Delta$ and that α is a free variable appearing in $\Gamma \longrightarrow \Delta$. Further suppose V is a $\Sigma_0^{1,b}$–abstract. Let $\Gamma(V)$ and $\Delta(V)$ denote the cedents obtained by substituting V for every occurrence of α in the formulae in Γ and Δ. Then $\Gamma(V) \longrightarrow \Delta(V)$ has a free cut free $\tilde{U}_2^i$–proof (respectively, $\tilde{V}_2^i$–proof).

Proof:

(a) is easily proved by induction of the complexity of A.

(b) is proved by exactly the same proof as the free cut elimination theorem for first order logic. We omit the proof here, the reader may refer to Takeuti [28], pp. 22-29, 112.

To prove (c), we may assume without loss of generality that P is in free variable normal form and that V has no bound variables in common with P. Let $P(V)$ denote the proof obtained from P by substituting V for every occurrence of α in formulae in P. It is easy to see by examining the allowable inferences that every inference in $P(V)$ is a valid inference of $\tilde{U}_2^i$

(respectively, $\tilde{V}_2^i$). In particular, (a) guarantees that $\Sigma_i^{1,b}$-*PIND* or $\Sigma_i^{1,b}$-*IND* , and $\Sigma_0^{1,b}$-*CR* inferences are still valid after the substitution of V for α.

However, $P(V)$ may fail to be a proof in that there may be initial sequents of $P(V)$ of the form

$$s_1=t_1, \ldots, s_n=t_n, A(s_1, \ldots, s_n) \longrightarrow A(t_1, \ldots, t_n)$$

where $V=\{\vec{x}\}A(\vec{x})$. However, sequents of this form are easy to prove without free cuts. So we merely tack onto $P(V)$ free cut free proofs of these initial sequents and thus obtain a proof Q of $\Gamma(V) \longrightarrow \Delta(V)$.

Q is not necessarily free cut free, as Q may contain free cuts with principal formula $A(\vec{s})$. But since V is a first order abstract, every free cut inference in Q has a first order principal formula. Hence, by (b), there is a free cut free proof of $\Gamma(V) \longrightarrow \Delta(V)$. □

Theorem 10: (Cut Elimination Theorem). Let P be a $\tilde{U}_2$-proof or a $\tilde{V}_2$-proof. Then there is a proof P^* in the same theory such that P^* has the same endsequent as P and there are no free cuts in P^*. Furthermore, each principal formula of an induction inference in P^* is a substitution instance of a principal formula of an induction inference in P and each principal abstract of a comprehension inference in P^* is a substitution instance of a principal abstract of a comprehension inference in P. Hence, for all $i \geq 0$, if P is a $\tilde{U}_2^i$- (or $\tilde{V}_2^i$-) proof then so is P^*.

Proof: We shall modify Takeuti's exposition on pages 22-29, 112, 143-144 of [28]. The reader should have [28] available as he reads the proof.

Following [28], we define the ***grade*** of a formula A to be the number of logical symbols in A. The ***level*** of A is the number of second order quantifiers in A.

A ***mix*** inference with principal formula A is of the form

$$\frac{\Gamma \longrightarrow \Delta \qquad \Pi \longrightarrow \Lambda}{\Gamma, \Pi^* \longrightarrow \Delta^*, \Lambda}$$

where Π^* and Δ^* are obtained from Π and Δ by removing all occurrences of A. A mix inference is ***free*** iff all of the occurrences of A in Δ and Π are free. Since a mix inference and a cut inference are so similar, it suffices to prove Theorem 10 for proofs which use mix inferences instead of cut inferences.

Suppose P is a proof whose last inference is a mix with principal formula A as shown above. Define the ***distance*** of a sequent in P to be the number of inferences separating it from the endsequent of P. The ***right rank*** of P is defined to be the maximum distance of a sequent containing a direct ancestor of an occurrence of A in the cedent Π. The ***left rank*** of P is the maximum distance of a sequent containing a direct ancestor of an occurrence of A in the cedent

Δ. The *rank* of P is the sum of the right rank and left rank.

It suffices to consider P with a single mix inference as the last inference. The proof is by ordinal induction on

$$ord(P) = \omega^2 \cdot \text{level}(P) + \omega \cdot \text{grade}(P) + \text{rank}(P)$$

where level(P) and grade(P) are the level and grade of the principal formula of the final mix of P, and ω is the first infinite ordinal.

Thus it suffices to show that if P is a proof with no free mixes except for the final inference of P and if Theorem 10 is satisfied for all proofs P' with $ord(P') < ord(P)$, then P satisfies Theorem 10. We modify the proof of Lemma 5.4 of Takeuti [28]:

Case (1): rank(P)=2.

Cases (1.1)-(1.5.ii): Similar to pages 24-27 of [28].

Case (1.5.iii): Suppose $A=(\forall\phi)B(\phi)$ and the last inferences of P are

$$\dfrac{\dfrac{\Gamma \longrightarrow \Delta, B(\alpha)}{\Gamma \longrightarrow \Delta, (\forall\phi)B(\phi)} \qquad \dfrac{B(V), \Pi \longrightarrow \Lambda}{(\forall\phi)B(\phi), \Pi \longrightarrow \Lambda}}{\Gamma, \Pi \longrightarrow \Delta, \Lambda}$$

where V is a $\Sigma_0^{1,b}$–abstract, and since rank(P)=2 the indicated occurrences of $(\forall\phi)B(\phi)$ are the only ones. By Lemma 9(c), we can obtain a free mix free proof of $\Gamma \longrightarrow \Delta, B(V)$ from the free mix free proof of $\Gamma \longrightarrow \Delta, B(\alpha)$. Thus we have a proof Q such that the only free mix in Q is its last inference:

$$\dfrac{\Gamma \longrightarrow \Delta, B(V) \quad B(V), \Pi \longrightarrow \Lambda}{\Gamma, \Pi^{\#} \longrightarrow \Delta^{\#}, \Lambda}$$

where $\Pi^{\#}$ and $\Delta^{\#}$ are Π and Δ minus all occurrences of $B(V)$.

By the induction hypothesis, there is a free mix free proof Q^* of $\Gamma, \Pi^{\#} \longrightarrow \Delta^{\#}, \Lambda$ since $ord(Q) < ord(P)$. By adding weak inferences to the end of Q^* we obtain the desired proof P^*.

Case (1.5.iv): Suppose $A=(\exists\phi)B(\phi)$. This case is very similar to Case (1.5.iii).

Case (2): rank(P)>2.

Case (2.1): The right rank of P is >1.

Cases (2.1.1)-(2.1.3.ii): Similar to [28].

Case (2.1.3.iii): Suppose $A{=}(\forall\phi)B(\phi)$, V is a $\Sigma_0^{1,b}$-abstract and the last inferences of P are:

$$\dfrac{\Gamma\longrightarrow\Delta \qquad \dfrac{B(V),\Pi\longrightarrow\Lambda}{(\forall\phi)B(\phi),\Pi\longrightarrow\Lambda}}{\Gamma,\Pi^*\longrightarrow\Delta^*,\Lambda}$$

where now Δ and Π, but not Γ, may contain occurrences of $(\forall\phi)B(\phi)$. The cedents Π^* and Δ^* are Π and Δ minus all occurrences of $(\forall\phi)B(\phi)$. Modify the end of P to obtain a proof P_1 which ends as

$$\dfrac{\Gamma\longrightarrow\Delta \qquad B(V),\Pi\longrightarrow\Lambda}{\Gamma,B(V),\Pi^*\longrightarrow\Delta^*,\Lambda}$$

The right rank of P_1 is one less than the right rank of P so by the induction hypothesis there is a free mix free proof P_2 of the endsequent of P_1. Now consider the proof which ends

$$\dfrac{\Gamma\longrightarrow\Delta \qquad \dfrac{B(V),\Gamma,\Pi^*\longrightarrow\Delta^*,\Lambda}{(\forall\phi)B(\phi),\Gamma,\Pi^*\longrightarrow\Delta^*,\Lambda}}{\Gamma,\Gamma,\Pi^*\longrightarrow\Delta^*,\Delta^*,\Lambda}$$

The right rank of this is one, so by the induction hypothesis and some exchanges and contractions we obtain a free mix free proof of $\Gamma,\Pi^*\longrightarrow\Delta^*,\Lambda$.

The rest of the cases are similar.

Q.E.D. □

9.5. $\Sigma_1^{1,b}$-Defined Functions and $\Delta_1^{1,b}$-Defined Predicates.

The second order theories of Bounded Arithmetic are in many respects analogous to the first order theories S_2^i and T_2^i. One of the most fundamental properties of second order Bounded Arithmetic is that new function and predicate symbols may be introduced into the language of Bounded Arithmetic; under certain conditions, these new function and predicate symbols may be used freely in the principal formulae of induction axioms and comprehension axioms.

Definition: Let R be a second order theory of Bounded Arithmetic. Suppose $A(\vec{x},y)$ is a $\Sigma_1^{1,b}$-formula with all free variables indicated and that

$$(1)\ R \vdash (\forall \vec{x})(\exists y \le t) A(\vec{x},y)$$
$$(2)\ R \vdash (\forall \vec{x})(\forall y)(\forall z)(A(\vec{x},y) \wedge A(\vec{x},z) \supset y=z).$$

Then we say that R can $\Sigma_1^{1,b}$-define the function f such that $\mathbf{N} \models (\forall \vec{x}) A(\vec{x}, f(\vec{x}))$.

The $\Sigma_1^{1,b}$-defined function symbols and the $\Delta_1^{1,b}$-defined predicate symbols play the same role in the second order theories of Bounded Arithmetic as the Σ_1^b-defined function symbols and the Δ_1^b-defined predicate symbols do in the first order theories S_2^i and T_2^i. In particular, the analogues of Theorems 2.2, 2.3 and 2.4 hold for second order Bounded Arithmetic.

Definition: Let $\vec{f}$ and $\vec{p}$ be new function and predicate symbols. The sets $\Sigma_i^{1,b}(\vec{f},\vec{p})$ and $\Pi_i^{1,b}(\vec{f},\vec{p})$ are sets of bounded formulae in the language of second order Bounded Arithmetic plus the symbols $\vec{f}$ and $\vec{p}$. These sets are defined by counting alternations of second order quantifiers ignoring the first order, bounded quantifiers.

Theorem 11: Let R be a second order theory of Bounded Arithmetic. Suppose R can $\Sigma_1^{1,b}$-define each of the functions $\vec{f}$ and can $\Delta_1^{1,b}$-define each of the predicates $\vec{p}$. Let R^* be the theory obtained from R by adjoining the new symbols $\vec{f}$ and $\vec{p}$ and their defining axioms. Then, if $i>0$ and B is a $\Sigma_i^{1,b}(\vec{f},\vec{p})$- (or a $\Pi_i^{1,b}(\vec{f},\vec{p})$-) formula, then there is a formula $B^* \in \Sigma_i^{1,b}$ (or $\Pi_i^{1,b}$, respectively) such that $R^* \vdash B^* \leftrightarrow B$.

The proof of Theorem 11 is similar to the proofs of Theorems 2.2 and 2.4.

Definition: Let R be a theory of Bounded Arithmetic and let $\vec{f}$ be a vector of defined function symbols of R and $\vec{p}$ be a vector of defined predicate symbols. Then $R(\vec{f},\vec{p})$ is the conservative extension of R obtained by enlarging the language to include $\vec{f}$ and $\vec{p}$ and including the defining axioms for these symbols.

Corollary 12: $(i \ge 1)$:

(a) Let $\vec{f}$ be a vector of $\Sigma_1^{1,b}$-defined function symbols of U_2^i (respectively, V_2^i) and let $\vec{p}$ be a vector of $\Delta_1^{1,b}$-defined predicate symbols of U_2^i (respectively, V_2^i). Then $U_2^i(\vec{f},\vec{p})$ (respectively, $V_2^i(\vec{f},\vec{p})$) has as theorems the $\Sigma_i^{1,b}(\vec{f},\vec{p})$-*PIND* axioms (respectively, the $\Sigma_i^{1,b}(\vec{f},\vec{p})$-*IND* axioms).

(b) Let $\vec{f}$ be a vector of $\Sigma_1^{1,b}$-defined function symbols of $\tilde{U}_2^i$ (respectively, $\tilde{V}_2^i$) and let $\vec{p}$ be a vector of $\Delta_1^{1,b}$-defined predicate symbols of $\tilde{U}_2^i$ (respectively, $\tilde{V}_2^i$). Then $\tilde{U}_2^i(\vec{f},\vec{p})$ (respectively, $\tilde{V}_2^i(\vec{f},\vec{p})$) has as theorems the $\tilde{\Sigma}_i^{1,b}(\vec{f},\vec{p})$-*PIND* axioms (respectively, the $\tilde{\Sigma}_i^{1,b}(\vec{f},\vec{p})$-*IND* axioms).

(c) Let $\vec{f}$ be a vector of $\Sigma_1^{1,b}$-defined function symbols of $\mathring{U}_2^i$ (respectively, $\mathring{V}_2^i$) and let $\vec{p}$ be a vector of $\Delta_1^{1,b}$-defined predicate symbols of $\mathring{U}_2^i$ (respectively, $\mathring{V}_2^i$). Then $\mathring{U}_2^i(\vec{f},\vec{p})$

(respectively, $\mathring{V}_2^i(\vec{f},\vec{p})$) has as theorems the $\tilde{\Sigma}_i^{1,b}$*-PIND* $(\vec{f},\vec{p})$ axioms (respectively, the $\tilde{\Sigma}_i^{1,b}$*-IND* $(\vec{f},\vec{p})$ axioms) and the $\tilde{\Delta}_1^{1,b}(\vec{f},\vec{p})$*-CA* axioms.

Corollary 12 tells us that $\Sigma_1^{1,b}$-defined function symbols and $\Delta_1^{1,b}$-defined predicate symbols may be used freely in the principal formulae of induction inferences. Furthermore, if we are working in the theory $\mathring{U}_2^i$ or $\mathring{V}_2^i$ we may use such function and predicate symbols freely in principal abstracts of comprehension inferences.

The next two theorems give an application of $\Delta_1^{1,b}$-comprehension to show that $\mathring{U}_2^1$ and $\mathring{V}_2^1$ can define the iteration of $\Delta_1^{1,b}$-defined predicates. It is an open question whether these theorems hold for the theories U_2^1 and V_2^1.

Theorem 13: Let $A(a,\vec{c},\vec{\gamma})$ and $B(a,b,\vec{c},\alpha_1^1,\vec{\gamma})$ be $\Delta_1^{1,b}$-formulae of $\mathring{U}_2^1$, where α_1^1 is a unary predicate variable. Let $t(b,\vec{c})$ be a term which contains only the free variables b and $\vec{c}$. Then the predicate $K(a,b,\vec{c},\vec{\gamma})$ which satisfies

$$K(a,b,\vec{c},\vec{\gamma}) \iff \begin{cases} A(a,\vec{c},\vec{\gamma}) & \text{if } b=0 \text{ and } a\le t(b,\vec{c}) \\ 0=1 & \text{if } a>t(b,\vec{c}) \\ B(a,b,\vec{c},\{x\}K(x,\lfloor\frac{1}{2}b\rfloor,\vec{c},\vec{\gamma}),\vec{\gamma}) & \text{otherwise} \end{cases}$$

is $\Delta_1^{1,b}$-definable by $\mathring{U}_2^1$.

Proof: The idea, of course, is to define $K(a,b,\vec{c},\vec{\gamma})$ by induction on the length of b. Let $B^*(a,b,\vec{c},\alpha_1^2,\vec{\gamma})$ be the formula

$$B(a,b,\vec{c},\{x\}[\alpha_1^2(x,\lfloor\tfrac{1}{2}b\rfloor)\wedge x\le t(\lfloor\tfrac{1}{2}b\rfloor,\vec{c})],\vec{\gamma})$$

and let $D(u,\phi)$ be the formula

$$(\forall y<2^{|u|})(\forall x\le t(y,\vec{c}))[\phi(x,y)\leftrightarrow((B^*(x,y,\vec{c},\phi,\vec{\gamma})\wedge y\neq 0)\vee(A(x,\vec{c},\vec{\gamma})\wedge y=0))].$$

It is easy to see that

$$\mathring{U}_2^1\vdash(\exists\phi)D(0,\phi)$$

and

$$\mathring{U}_2^1\vdash(\exists\phi)D(\lfloor\tfrac{1}{2}u\rfloor,\phi)\supset(\exists\phi)D(u,\phi).$$

Hence, by $\Sigma_1^{1,b}$*-PIND* , $\mathring{U}_2^1\vdash(\forall z)(\exists\phi)D(z,\phi)$. It is also not difficult to use $\Sigma_1^{1,b}$*-PIND* to prove that

$$\mathring{U}_2^1 \vdash D(u,\phi) \wedge D(u,\psi) \supset (\forall y < 2^{|u|})(\forall x \le t(y,\vec{c}))(\psi(x,y) \leftrightarrow \phi(x,y)).$$

Hence, $\mathring{U}_2^1$ can $\Delta_1^{1,b}$–define K by

$$K(a,b,\vec{c},\vec{\gamma}) \iff a \le t(b,\vec{c}) \wedge (\exists\phi)(D(b,\phi) \wedge \phi(a,b))$$

and by the provably equivalent

$$K(a,b,\vec{c},\vec{\gamma}) \iff (\forall\phi)[D(b,\phi) \supset \phi(a,b)].$$

Q.E.D. □

Note that it is important to the proof of Theorem 13 that the support of K was bounded by the requirement that $a \le t(b,\vec{c})$; otherwise the formula $D(u,\phi)$ could not be bounded. Theorem 13 is false without this restriction.

A similar use of $\Delta_1^{1,b}$–comprehension can be made by $\mathring{V}_2^1$:

Theorem 14: Let $A(a,\vec{c},\vec{\gamma})$ and $B(a,b,\vec{c},\alpha_1^1,\vec{\gamma})$ be $\Delta_1^{1,b}$–formulae of $\mathring{V}_2^1$ where α_1^1 is a unary predicate variable. Let $t(b,\vec{c})$ be a term with the free variables indicated. Then the predicate $K(a,b,\vec{c},\vec{\gamma})$ which satisfies

$$K(a,b,\vec{c},\vec{\gamma}) \iff \begin{cases} A(a,\vec{c},\vec{\gamma}) & \text{if } b=0 \text{ and } a \le t(b,\vec{c}) \\ 0=1 & \text{if } a > t(b,\vec{c}) \\ B(a,b,\vec{c},\{x\}K(x,b \dot{-} 1,\vec{c},\vec{\gamma}),\vec{\gamma}) & \text{otherwise} \end{cases}$$

is $\Delta_1^{1,b}$–definable in $\mathring{V}_2^1$.

The $\Delta_1^{1,b}$–formulae are in some respects more akin to the $\Delta_0^{1,b}$–formulae than to the $\Sigma_1^{1,b}$– and $\Pi_1^{1,b}$–formulae. For example, we have the following theorem:

Theorem 15:

(a) The $\Delta_1^{1,b}$–*IND* axioms and the $\Delta_1^{1,b}$–*MIN* axioms are theorems of U_2^1 and V_2^1.

(b) The $\Delta_1^{1,b}$–*IND* axioms and the $\Delta_1^{1,b}$–*MIN* axioms are theorems of $\mathring{U}_2^1$ and $\mathring{V}_2^1$.

Proof: It is obvious that the $\Delta_1^{1,b}$–*IND* axioms are theorems of V_2^1. The fact that the $\Delta_1^{1,b}$–*IND* axioms are theorems of U_2^1 is proved just like Theorem 2.22.

Now we claim that the $\Delta_1^{1,b}$–*MIN* axioms follow from the $\Delta_1^{1,b}$–*IND* axioms. Indeed, the minimization axiom for a $\Delta_1^{1,b}$–formula A can be proved by using induction on the $\Delta_1^{1,b}$–formula $(\forall y \le x)(\neg A(y))$. This proves (a).

(b) is proved similarly.

Q.E.D. □

9.6. $\Sigma_1^{1,b}$-Replacement.

The $\Sigma_i^{1,b}$–replacement axioms of second order Bounded Arithmetic are analogous to the Σ_i^b–replacement axioms of first order Bounded Arithmetic. The $\Sigma_i^{1,b}$–replacement axioms provide us with the ability to interchange the order of second order quantifiers and first order bounded quantifiers.

To state the definition of the $\Sigma_i^{1,b}$–replacement axioms, we need first to define an analogue of the Gödel beta function which operates on predicates.

Definition: Let α be a second order unary predicate variable. We write $\boldsymbol{\beta}(b,\alpha)$ as an abbreviation for the atomic abstract $\{x\}\alpha(<b,x>)$.

The motivation behind this definition of $\boldsymbol{\beta}$ is that it can be used as a Gödel beta function operating on predicate variables. One simple application of $\boldsymbol{\beta}$ is as a pairing function. Thus, we can think of the predicate variable α coding the two predicates $\beta=\boldsymbol{\beta}(1,\alpha)$ and $\gamma=\boldsymbol{\beta}(2,\alpha)$. Conversely, given two unary predicates variables β and γ, the $\Sigma_0^{1,b}$–comprehension axioms guarantee the existence of a predicate α such that

$$\alpha(x) \iff \begin{cases} \beta(y) & \text{if } x=<1,y> \\ \gamma(y) & \text{if } x=<2,y> \\ 0=1 & \text{otherwise} \end{cases}$$

and thus $\beta=\boldsymbol{\beta}(1,\alpha)$ and $\gamma=\boldsymbol{\beta}(2,\alpha)$.

Definition: We write $<\beta,\gamma>$ to denote the predicate α defined as above. More precisely, $<\gamma_1,\gamma_2>$ is an abbreviation for the abstract

$$\{x\}\left[(\exists z \le x)((x=<1,z>\wedge\gamma_1(z))\vee(x=<2,z>\wedge\gamma_2(z)))\right].$$

As Theorem 16 below shows, $\boldsymbol{\beta}$ can be used for more sophisticated purposes than just as a pairing function.

Definition: The $\Sigma_i^{1,b}$–replacement axioms are the formulae of the form

$$(\forall x \le t)(\exists \phi)A(x,\phi) \leftrightarrow (\exists \phi)(\forall x \le t)A(x,\boldsymbol{\beta}(x+1,\phi))$$

where t is any term, ϕ is a unary predicate variable, and A is any $\Sigma_i^{1,b}$–formula. Other first and second order free variables may appear in A as parameters.

Theorem 16: Let $i \ge 1$. Then $\Sigma_i^{1,b}$–replacement axioms are theorems of both U_2^i and V_2^i.

Proof: Let $A(b,\alpha)$ be a $\Sigma_i^{1,b}$–formula. Since V_2^i is a stronger theory than U_2^i (by Proposition 4), it will suffice to show that U_2^i proves the replacement axiom for A.

One direction is easy:

$$U_2^1 \vdash (\exists \phi)(\forall x \le t)A(x,\boldsymbol{\beta}(x+1,\phi)) \supset (\forall x \le t)(\exists \phi)A(x,\phi).$$

The other direction is more tricky. Let D be the formula $(\forall x \le t)(\exists \phi)A(x,\phi)$. Let $B(c)$ be the formula

$$(\forall y < 2^{|t| \dot{-} c})(\exists \phi)(\forall z < 2^{\min(|t|,c)})[y \cdot 2^{\min(|t|,c)} + z \le t \supset A(y \cdot 2^{\min(|t|,c)} + z, \boldsymbol{\beta}(z+1,\phi))].$$

Then it is obvious that $U_2^i \vdash D \supset B(0)$. Also it is straightforward to prove that $U_2^i \vdash D \wedge B(c) \supset B(c+1)$ by use of the $\Sigma_0^{1,b}$–comprehension axioms.

Since B is a $\Sigma_i^{1,b}$–formula, $U_2^i \vdash D \supset B(|t|)$ follows from $\Sigma_i^{1,b}$–*LIND*. Finally, it is clear that

$$U_2^i \vdash B(|t|) \leftrightarrow (\exists \phi)(\forall x \le t)A(x,\boldsymbol{\beta}(x+1,\phi)).$$

Hence the theorem is proved.

Q.E.D. □

Two more meta-predicates which are useful when used in conjunction with $\boldsymbol{\beta}$ are ARY_k and $DEARY_k$.

Definition: Let α be a second order unary predicate variable. We write $ARY_k(\alpha)$ as an abbreviation for the abstract $\{x_1, \ldots, x_k\}\alpha(<x_1, \ldots, x_k>)$.

Let γ be a k-ary predicate variable. We write $DEARY_k(\gamma)$ for the abstract $\{x\}\gamma(\beta(1,x), \ldots, \beta(k,x))$.

Hence $ARY_k(DEARY_k(\gamma))$ is the same predicate as γ. However, $DEARY_k(ARY_k(\alpha))$ is not in general the same as α.

As an example of how ARY_k and $DEARY_k$ can be used, consider the following more general form of the $\Sigma_i^{1,b}$–replacement:

$$(\forall x\leq t)(\exists\phi_1^k)A(x,\phi_1^k)\leftrightarrow(\exists\phi_1^1)(\forall x\leq t)A(x,ARY_k(\boldsymbol{\beta}(x+1,\phi_1^1)))$$

where ϕ_1^1 and ϕ_1^k are unary and k-ary, respectively. Of course this more general form of the $\Sigma_i^{1,b}$–replacement axiom is a consequence of the less general form presented above.

Corollary 17: Let $i\geq 1$. If A is a $\Sigma_i^{1,b}$–formula then there is a formula B of the form $(\exists\phi)C$ such that C is a $\Pi_{i-1}^{1,b}$–formula and such that U_2^i and V_2^i prove that A is equivalent to B.

Proof: By Lemma 6 we may assume without loss of generality that A is a $\tilde{\Sigma}_i^{1,b}$–formula. Now we may use prenex operations and the $\Sigma_i^{1,b}$–replacement axioms to transform A into the provably equivalent form

$$(\exists\phi_1)\cdots(\exists\phi_n)D(\phi_1,\ldots,\phi_n)$$

with $D\in\Pi_{i-1}^{1,b}$. The n second order existential quantifiers may be combined by use of the $\boldsymbol{\beta}$ function, giving B equal to

$$(\exists\phi)D(\boldsymbol{\beta}(1,\phi),\ldots,\boldsymbol{\beta}(n,\phi)).$$

Q.E.D. □

9.7. Cut Elimination in the Presence of $\Delta_1^{1,b}$-Comprehension.

In this section we investigate cut elimination theorems for $\hat{U}_2^i$ and $\hat{V}_2^i$. Although Gentzen's free cut elimination theorem holds for these theories, the proof is quite difficult and non-constructive. For our purposes, it will be sufficient to show that certain conservative extensions of $\hat{U}_2^i$ and $\hat{V}_2^i$ satisfy cut elimination.

One difficulty with proving the cut elimination theorem for $\hat{U}_2^i$ and $\hat{V}_2^i$ is that it is possible for $A(\alpha)$ to be a $\Sigma_i^{1,b}$–formula and U to be a $\Delta_1^{1,b}$–abstract and yet $A(U)$ is not a $\Sigma_i^{1,b}$–formula. Thus Lemma 9(c) is not readily provable for $\hat{U}_2^i$ and $\hat{V}_2^i$ when V is a $\Delta_1^{1,b}$–abstract.

A second and more serious difficulty arises when we try to prove the cut elimination theorem by induction on $ord(P)$ as in the proof of Theorem 10. In Case (1.5.iii) we transformed a mix inference with principal formula $(\forall\phi)B(\phi)$ to one with principal formula $B(V)$. Now if V

is merely a $\Delta_1^{1,b}$–abstract it is quite likely that the level of $B(V)$ is not less than the level of $(\forall\phi)B(\phi)$. However, without decreasing the level of the mix inference we can not apply the induction hypothesis in the proof by induction on $ord(P)$.

To circumvent these difficulties we shall define below theories $\mathring{U}_2^i(\delta)$ and $\mathring{V}_2^i(\delta)$ by enlarging the languages of $\mathring{U}_2^i$ and $\mathring{V}_2^i$. It will turn out that the constructive proof of the cut elimination used above in §9.4 can be extended to these expanded theories $\mathring{U}_2^i(\delta)$ and $\mathring{V}_2^i(\delta)$.

Definition: A *relational* δ is a predicate which acts on integers and predicates. More precisely, a k_0-ary relational δ is a subset of

$$\mathbf{N}^{k_0} \times \omega_{k_1}^1 \times \cdots \times \omega_{k_n}^1$$

where $n \geq 0$ and each $k_i \geq 1$ and ω_k^1 denotes the set of all k-ary predicates on the natural numbers.

Definition: Let R be a second order theory of Bounded Arithmetic. A relational δ is ***introduced by a*** $\Delta_1^{1,b}$***–definition*** in R iff the following hold:

(1) $A(\vec{a},\vec{\alpha})$ is a $\Sigma_1^{1,b}$–formula, $B(\vec{a},\vec{\alpha})$ is a $\Pi_1^{1,b}$–formula and $\vec{a}$ and $\vec{\alpha}$ indicate all of the free variables in A and B.

(2) $R \vdash A(\vec{a},\vec{\alpha}) \leftrightarrow B(\vec{a},\vec{\alpha})$.

(3) The defining equation for δ is

$$\delta(\vec{a},\vec{\alpha}) \iff A(\vec{a},\vec{\alpha}).$$

We will say that δ is $\Delta_1^{1,b}$–defined by R if the above holds and we write R_δ to denote the theory R enlarged to include the new symbol δ and its two defining equations:

(a) $\delta(\vec{a},\vec{\alpha}) \longrightarrow A(\vec{a},\vec{\alpha})$

(b) $A(\vec{a},\vec{\alpha}) \longrightarrow \delta(\vec{a},\vec{\alpha})$

These two defining equations are valid initial sequents of the natural deduction system for R_δ.

Definition: The theory $\mathring{U}_2^i(\delta)$ is the following natural deduction system:

(1) The *BASIC* axioms are initial sequents of $\mathring{U}_2^i(\delta)$. Also, logical axioms and equality axioms are valid initial sequents of $\mathring{U}_2^i(\delta)$.

(2) $\Sigma_i^{1,b}(\delta)$ and $\Pi_i^{1,b}(\delta)$ are the sets of formulae of the language of $\mathring{U}_2^i(\delta)$ defined in the usual way by counting alternations of bounded quantifiers, ignoring sharply bounded quantifiers.

(3) If $A \in \Sigma_1^{1,b}(\delta)$ and $B \in \Pi_1^{1,b}(\delta)$ and $R \vdash A(\vec{a},\vec{\alpha}) \leftrightarrow B(\vec{a},\vec{\alpha})$, then the relational δ defined by

$$\delta(\vec{a},\vec{\alpha}) \iff A(\vec{a},\vec{\alpha})$$

is a symbol of the language of $\mathring{U}_2^i(\delta)$. The two defining equations for δ are initial sequents of the natural deduction system for $\mathring{U}_2^i(\delta)$.

(4) The $\Sigma_i^{1,b}(\delta)$-*PIND* inferences are valid inferences of $\mathring{U}_2^i(\delta)$.

(5) The $\Sigma_0^{1,b}(\delta)$-*CR* comprehension inferences are valid inferences of $\mathring{U}_2^i(\delta)$.

$\mathring{U}_2(\delta)$ is the theory $\bigcup_i \mathring{U}_2^i(\delta)$.

Definition: $\mathring{V}_2^i(\delta)$ and $\mathring{V}_2(\delta)$ are defined similarly to $\mathring{U}_2^i(\delta)$ and $\mathring{U}_2(\delta)$ except using $\Sigma_i^{1,b}(\delta)$-*IND* instead of $\Sigma_i^{1,b}(\delta)$-*PIND*.

Definition: Let R be one of the theories $\mathring{U}_2^i(\delta)$, $\mathring{U}_2(\delta)$, $\mathring{V}_2^i(\delta)$ or $\mathring{V}_2(\delta)$. A formula A is $\Delta_1^{1,b}(\delta)$ with respect to R iff there is a $\Sigma_1^{1,b}(\delta)$-formula B and a $\Pi_1^{1,b}(\delta)$-formula C such that $R \vdash A \leftrightarrow B$ and $R \vdash A \leftrightarrow C$.

So, in effect, $\mathring{U}_2^i(\delta)$ and $\mathring{V}_2^i(\delta)$ are the same as the theories $\mathring{U}_2^i$ and $\mathring{V}_2^i$ except that all the $\Delta_1^{1,b}(\delta)$-defined relationals are included in the language and only $\Sigma_0^{1,b}(\delta)$-*CR* comprehension is allowed.

Proposition 18:

(a) $\mathring{U}_2^i(\delta)$ is a conservative extension of $\mathring{U}_2^i$.

(b) $\mathring{V}_2^i(\delta)$ is a conservative extension of $\mathring{V}_2^i$.

Proof:

(a) We begin by showing that $\mathring{U}_2^i(\delta)$ is an extension of $\mathring{U}_2^i$. For this it suffices to show that $\Delta_1^{1,b}$-comprehension is a derived rule of $\mathring{U}_2^i(\delta)$. So suppose $A \in \Sigma_1^{1,b}$, $B \in \Pi_1^{1,b}$ and $\mathring{U}_2^i(\delta) \vdash A \leftrightarrow B$. Let V be the abstract $\{\vec{x}\}A(\vec{x},\vec{b},\vec{\alpha})$ where $\vec{x}$, $\vec{b}$ and $\vec{\alpha}$ indicate all the free variables of A and suppose that $\mathring{U}_2^i(\delta)$ proves the sequent

$$\Gamma \longrightarrow \Delta, F(V).$$

Let δ be the relational of $\mathring{U}_2^i(\delta)$ which is $\Delta_1^{1,b}$-defined by

$$\delta(\vec{a},\vec{b},\vec{\alpha}) \iff A(\vec{a},\vec{b},\vec{\alpha}).$$

Let V_δ be the abstract $\{\vec{x}\}\delta(\vec{x},\vec{b},\vec{\alpha})$. Then there is a $\mathring{U}_2^i(\delta)$–proof which ends

$$\cfrac{\cfrac{\Gamma \longrightarrow \Delta, F(V) \qquad F(V) \longrightarrow F(V_\delta)}{\Gamma \longrightarrow \Delta, F(V_\delta)}}{\Gamma \longrightarrow \Delta, (\exists\phi)F(\phi)}$$

since V_δ is a $\Sigma_0^{1,b}(\delta)$–abstract (in fact, it is an atomic abstract).

Hence $\mathring{U}_2^i(\delta)$ is an extension of $\mathring{U}_2^i$. The fact that $\mathring{U}_2^i(\delta)$ is conservative over $\mathring{U}_2^i$ is proved just like Corollary 12(c).

(b) is proved similarly to (a).

Q.E.D. □

Because of the way we have defined the languages of $\mathring{U}_2^i(\delta)$ and $\mathring{V}_2^i(\delta)$ there will exist formulae $F(\alpha)$ such that $F(V)$ is not defined for V an arbitrary abstract. In particular, if F is $\delta(\alpha)$ for some relational δ, then $\delta(V)$ is not a formula and $F(V)$ is not defined. Thus we only allow $\Sigma_0^{1,b}(\delta)$–CR comprehension to be applied in those cases of the form

$$\frac{\Gamma \longrightarrow \Delta, F(V)}{\Gamma \longrightarrow \Delta, (\exists\phi)F(\phi)}$$

where $F(\alpha)$ is a formula such that $F(V)$ is defined. Of course, $F(V)$ is defined iff α is not an argument to any $\Delta_1^{1,b}$–defined relational in $F(\alpha)$.

We shall also need the capability to substitute a $\Delta_1^{1,b}$–abstract V for α in an arbitrary formula $F(\alpha)$. Accordingly, we make the following definition:

Definition: Let R be one of the theories $\mathring{U}_2^i(\delta)$ or $\mathring{V}_2^i(\delta)$. Let α be an n-ary predicate variable, $F(\alpha)$ be a formula in the language of R, and V be the abstract $\{y_1, \ldots, y_n\}A(\vec{y},\vec{b},\vec{\beta})$ where A is a $\Sigma_0^{1,b}(\delta)$–formula of R. Then $F[V]$ is defined by induction on the complexity of F:

(1) If α does not appear in F, then $F[V]$ is F.

(2) If $F(\alpha)$ is $\alpha(\vec{s})$, then $F[V]$ is $A(\vec{s},\vec{b},\vec{\beta})$.

(3) If $F(\alpha)$ is $\delta_C(\vec{c},\alpha,\vec{\gamma})$ where C is a $\Delta_1^{1,b}(\delta)$–formula of the theory R and δ_C is the relational with defining axiom

$$\delta_C(\vec{c},\alpha,\vec{\gamma}) \iff C(\vec{c},\alpha,\vec{\gamma})$$

then $F[V]$ is $\delta(\vec{c},\vec{\gamma})$ where δ is the relational defined by

$$\delta(\vec{c},\vec{\gamma}) \iff C[V](\vec{c},\vec{\gamma}).$$

Here $C[V]$ is the result of substituting V for α in C. Notice that since A is a $\Sigma_0^{1,b}(\delta)$-formula and C is $\Delta_1^{1,b}(\delta)$ with respect to R, so is $C[V]$.

(4) Suppose F is $\neg B$, $B\wedge C$, $B\vee C$ or $B\supset C$. Then $F[V]$ is $\neg B[V]$, $B[V]\wedge C[V]$, $B[V]\vee C[V]$ or $B[V]\supset C[V]$, respectively.

(5) Suppose $F(\alpha)$ is $(\forall x)B(\alpha)$ or $(\exists x)B(\alpha)$. If x appears in A, we obtain A' by renaming the variable x in A to avoid conflict of variables. Then $F[V]$ is $(\forall x)B[\{\vec{y}\}A'(\vec{y})]$ or $(\exists x)B[\{\vec{y}\}A'(\vec{y})]$, respectively.

(6) Suppose $F(\alpha)$ is $(\forall\phi)B(\alpha)$ or $(\exists\phi)B(\alpha)$. Since A has no second order quantifiers, the bound variable ϕ does not appear in A. So $F[V]$ is $(\forall\phi)B[\{\vec{y}\}A(\vec{y})]$ or $(\exists\phi)B[\{\vec{y}\}A(\vec{y})]$, respectively.

Definition: Let $A(\alpha_1,\ldots,\alpha_n,a_1,\ldots,a_m)$ be a formula where the α's and a's indicate all of the free variables in A. B is a ***substitution instance*** of A iff B is of the form $A[V_1,\ldots,V_n](t_1,\ldots,t_m)$ where each V_i is a $\Sigma_0^{1,b}(\delta)$-abstract and is substituted in for α_i and each t_i is a term substituted in for a_i.

The next lemma is analogous to Lemma 9. It will be exactly what we need to carry out the proof of the cut elimination theorem for $\mathring{U}_2^i(\delta)$ and $\mathring{V}_2^i(\delta)$.

Lemma 19: Let $i\geq 0$ and let R be one of the theories $\mathring{U}_2^i(\delta)$ or $\mathring{V}_2^i(\delta)$.

(a) If B is a $\Sigma_i^{1,b}(\delta)$-formula (respectively, a $\Pi_i^{1,b}(\delta)$-formula), then every substitution instance of B is a $\Sigma_i^{1,b}(\delta)$-formula (respectively, a $\Pi_i^{1,b}(\delta)$-formula).

(b) Suppose P is an R-proof of $\Gamma\longrightarrow\Delta$ and that every free cut in P has a first order formula as its principal formula. Then there is a free cut free R-proof of $\Gamma\longrightarrow\Delta$.

(c) Suppose P is a free cut free R-proof of $\Gamma\longrightarrow\Delta$ and that α is a free variable appearing in $\Gamma\longrightarrow\Delta$. Further suppose V is a $\Sigma_0^{1,b}(\delta)$-abstract. Let $\Gamma[V]$ and $\Delta[V]$ denote the cedents obtained by substituting V for every occurrence of α in the formulae in Γ and Δ. Then $\Gamma[V]\longrightarrow\Delta[V]$ has a free cut free R-proof.

Proof:

(a) is easily proved by induction on the complexity of A.

(b) is proved exactly like the free cut elimination theorem for first order logic (Theorem 4.3). Refer to Takeuti [28], pp. 22-29, 112 for details.

To prove (c), we may assume without loss of generality that P is in free variable normal form and that V has no bound variables in common with P. Let $P[V]$ denote the proof obtained from P by substituting V for every occurrence of α in formulae in P. It is easy to see by examining the allowable inferences that every inference in $P[V]$ is a valid inference of $\mathring{U}_2^i(\delta)$ (respectively, $\mathring{V}_2^i(\delta)$). In particular, (a) guarantees that $\Sigma_i^{1,b}(\delta)$-*PIND* or $\Sigma_i^{1,b}(\delta)$-*IND* and

$\Sigma_0^{1,b}(\delta)$–CR inferences are still valid after the substitution of V for α.

If P contains any initial sequents which are defining axioms for some relational δ, say

$$\delta(\alpha,\vec{a},\vec{\beta}) \longrightarrow C(\alpha,\vec{a},\vec{\beta})$$

then in $P[V]$ this initial sequent becomes

$$\delta^*(\vec{a},\vec{\beta}) \longrightarrow C[V](\vec{a},\vec{\beta})$$

where δ^* is $\delta[V]$. This is a defining axiom for δ^* and hence is a valid initial sequent.

However, $P[V]$ may fail to be a proof in that there may be initial sequents of $P[V]$ of the form

$$s_1=t_1,\ldots,s_n=t_n,A(s_1,\ldots,s_n) \longrightarrow A(t_1,\ldots,t_n)$$

where A is not atomic. However, sequents of this form are easy to prove without free cuts. So we merely tack onto $P[V]$ free cut free proofs of these initial sequents and thus obtain a proof Q of $\Gamma[V] \longrightarrow \Delta[V]$.

Q is not necessarily free cut free since Q may contain free cuts with principal formulae of the form $B[V]$ where B is atomic. But each $B[V]$ is first order and so by (b) there is a free cut free R–proof of $\Gamma[V] \longrightarrow \Delta[V]$. □

Theorem 20: Let R be one of the theories $\mathring{U}_2^i(\delta)$ or $\mathring{V}_2^i(\delta)$ where $i \geq 0$. Let P be an R–proof. Then there is an R–proof P^* such that P^* has the same endsequent as P and there are no free cuts in P^*. Furthermore, each principal formula of an induction inference in P^* is a substitution instance of a principal formula of an induction inference in P and each principal abstract of a comprehension inference in P^* is a substitution instance of a principal abstract of a comprehension inference in P.

Proof: The proof follows the proof of Theorem 10 (and Takeuti [28]) almost exactly. We define the order *ord*(P) of P as before and proceed by induction on the *ord*(P). The only difference is that in Case (1.5.iii) we use Lemma 19(c) instead of Lemma 9(c).

Q.E.D. □

Corollary 21: Let R be one of the theories $\mathring{U}_2^i(\delta)$ or $\mathring{V}_2^i(\delta)$ where $i \geq 1$. Suppose R proves the sequent $\Gamma \longrightarrow \Delta$ and that every formula in $\Gamma \cup \Delta$ is a $\Sigma_i^{1,b}(\delta)$–formula or a $\Pi_i^{1,b}(\delta)$–formula. Then there is an R–proof P of $\Gamma \longrightarrow \Delta$ such that every formula in P is in $\Sigma_i^{1,b}(\delta)$ or in $\Pi_i^{1,b}(\delta)$.

Proof: By Theorem 20 there is a free cut free proof P of $\Gamma \longrightarrow \Delta$. If A is the principal formula of a cut in P, then A must be a direct descendant of either a principal formula of an induction inference or of a formula in an initial sequent. In the first case, A must be a $\Sigma_i^{1,b}(\delta)$–formula since only $\Sigma_i^{1,b}(\delta)$–*PIND* (or $\Sigma_i^{1,b}(\delta)$–*IND*) inferences are allowed. In the second case, we claim that A is in $\Sigma_1^{1,b}(\delta)$. This is because each initial sequent must either (a) be an equality or *BASIC* axiom and contain only atomic formulae or (b) be a defining equation for a relational.

Now it is clear that every formula in P must be in $\Sigma_i^{1,b}(\delta)$ or $\Pi_i^{1,b}(\delta)$ since a formula can only be removed via a cut inference and no other kind of inference can reduce the alternations of second order quantifiers in a formula. In particular, note that since only $\Sigma_0^{1,b}(\delta)$–*CR* comprehension inferences are allowed, any comprehension inference of the form

$$\frac{\Gamma \longrightarrow \Delta, A(V)}{\Gamma \longrightarrow \Delta, (\exists \phi) A(\phi)}$$

will have $A \in \Sigma_i^{1,b}(\delta)$ if $(\exists \phi) A(\phi) \in \Sigma_i^{1,b}(\delta)$.

Q.E.D. □

Corollary 21 is exactly what we need to prove the main theorems of Chapter 10.

Chapter 10

Definable Functions of Second Order Bounded Arithmetic

This chapter investigates the question of what functions are $\Sigma_1^{1,b}$–definable in the second order theories U_2^1 and V_2^1 of Bounded Arithmetic. It turns out that a function f with polynomial growth rate is $\Sigma_1^{1,b}$–definable in U_2^1 (or in $\mathring{U}_2^1$) iff f is computable by a polynomial space bounded Turing machine, i.e., iff f is in PSPACE. In addition, f is $\Sigma_1^{1,b}$–definable in V_2^1 (or in $\mathring{V}_2^1$) iff f is computable by an exponential time bounded Turing machine, i.e., iff f is in EXPTIME.

10.1. EXPTIME functions are $\Sigma_1^{1,b}$-definable in V_2^1.

Definition: EXPTIME is the set of functions f of polynomial growth rate which can be computed by a Turing machine M_f such that there is a polynomial $p(\vec{n})$ so that the runtime of M_f on input $\vec{x}$ is always less than $2^{p(|\vec{x}|)}$.

Our definition of EXPTIME differs from the usual definition used by computer scientists. Usually EXPTIME is taken to be a set of predicates; however, we are using it as a set of functions with polynomial growth rate. We shall also talk about predicates being in EXPTIME: if P is a predicate, then we define P is in EXPTIME to mean that the characteristic function of P is in EXPTIME.

We shall also need the concept of exponential time functionals, which are defined analogously to the polynomial hierarchy of functionals of Chapter 1. Recall that ω_n^1 is equal to the set of n-ary predicates on the natural numbers.

Definition: Let $\phi_1, \ldots, \phi_r$ be predicate variables of a second order theory of Bounded Arithmetic, where each ϕ_i is k_i-ary. Then EXPTIME$(\phi_1, \ldots, \phi_r)$ is the uniform set of functionals f such that the following hold:

(1) f has polynomial growth rate.

(2) For some $k_f \geq 1$, f has domain

$$\mathbf{N}^{k_f} \times \omega_{k_1}^1 \times \cdots \times \omega_{k_r}^1.$$

(3) There is an oracle Turing machine M_f with r oracles such that for $1 \le i \le r$, the i-th oracle is k_i-ary and such that for all $\Omega_1, \ldots, \Omega_r$ with $\Omega_i \in \omega_{k_i}^1$,

$$f(\vec{x}, \Omega_1, \ldots, \Omega_r) = M_f(\vec{x}, \Omega_1, \ldots, \Omega_r)$$

where $M_f(\vec{x}, \Omega_1, \ldots, \Omega_r)$ denotes the value output by M_f on input $\vec{x}$ with oracles $\Omega_1, \ldots, \Omega_r$.

(4) For some polynomial $p(\vec{n})$, the runtime of $M_f(\vec{x}, \Omega_1, \ldots, \Omega_r)$ is less than $2^{p(|\vec{x}|)}$ for all $\vec{x}$ and all $\Omega_1, \ldots, \Omega_r$.

(5) For all $\vec{x}$ and $\Omega_1, \ldots, \Omega_r$, $M_f(\vec{x}, \Omega_1, \ldots, \Omega_r)$ uses no more than $p(|\vec{x}|)$ tape squares on each of its oracle tapes. Or equivalently, $M_f(\vec{x}, \Omega_1, \ldots, \Omega_r)$ only queries its oracles about $\Omega_i(\vec{y})$ for $\vec{y} < 2^{p(|\vec{x}|)}$.

We will also denote EXPTIME$(\phi_1, \ldots, \phi_r)$ by EXPTIME$(\omega_{k_1}^1, \ldots, \omega_{k_r}^1)$.

Condition (5) in the definition above is somewhat unusual in that it bounds the size of the oracle queries of M_f. This is, however, actually a very natural condition since it means that if $\Omega \in$EXPTIME and $M(x,\phi) \in$EXPTIME(ϕ) then $M(x,\Omega) \in$EXPTIME. Without condition (5) this would not necessarily be true.

Theorem 1: Let f be a function of polynomial growth rate in EXPTIME. Then f is $\Sigma_1^{1,b}$-definable in V_2^1.

Proof: Let us assume without loss of generality that f is a unary function and M is a single tape Turing machine which runs in time less that $2^{q(|\vec{x}|)}$ for each input x, where q is a suitable polynomial. Let the alphabet of M be Γ where the cardinality $|\Gamma|$ of Γ is at least 3, and suppose that the symbols "b" , "0" and "1" are included in Γ. Let the states of M be $q_0, \ldots, q_N$ with q_0 the initial state. We let \$ be a new symbol not in Γ. We assign arbitrarily Gödel numbers to the states q_i, the symbols in Γ and to "\$" ; we denote these Gödel numbers by $\ulcorner q_i \urcorner$, $\ulcorner b \urcorner$, $\ulcorner \$ \urcorner$, etc. Let n be the maximum number used as a Gödel number.

An ID (instantaneous description) is an encoding of a state of M and is a sequence

$$\ulcorner \$ \urcorner, \ulcorner \gamma_1 \urcorner, \ldots, \ulcorner \gamma_k \urcorner, \ulcorner q_i \urcorner, \ulcorner \gamma_{k+1} \urcorner, \ldots, \ulcorner \gamma_v \urcorner, \ulcorner \$ \urcorner$$

where each γ_i is in Γ and q_i is the current state of M, the current tape head position is at γ_{k+1}, and the \$'s denote the immovable ends of the tape. To encode ID's of M in the theory V_2^1, we shall use a second order function symbol ς^n with values less than or equal to n.

Let $Next_M(a_1,a_2,a_3,a_4,b)$ be a predicate which is true when a_1,a_2,a_3,a_4 codes four consecutive values of an *ID* for M and b is the value which replaces a_2 in the next ID of M. For example, b must equal a_2 unless a_1, a_2, or a_3 is a Gödel number of a state of M. When a_1,a_2,a_3,a_4 do not code valid consecutive values for an ID of M then $Next_M(a_1,a_2,a_3,a_4,b)$ is true iff $b=\ulcorner \$ \urcorner$. It is easy to see that $Next_M$ is Σ_1^b–definable in V_2^1 (in fact, $Next_M$ is easily seen to be Σ_1^b–definable in S_2^1.)

Define $r(x)$ to be equal to $2^{q(|x|)}+|x|+2$; then $r(x)$ is expressible by a term of Bounded Arithmetic. We can assume without loss of generality that on input x, each ID of M is of length exactly $r(x)$. We code the run of M on input x by the function ς^n so that for all $j\leq r(x)\cdot 2^{q(|x|)}$, $\varsigma^n(j)$ is equal to the $(Rem(j,r(x))+1)$-th number in the $(\lfloor j/r(x)\rfloor+1)$-th ID of the run of M on input x.

Accordingly, we define a predicate $Init_M$ as:

$$
\begin{aligned}
Init_M(\varsigma,x) \iff & (\forall i<r(x))[(i=1\supset\varsigma(i)=\ulcorner q_0\urcorner)\wedge\\
&\wedge(i=0\vee i=r(x)\dot{-}1\supset\varsigma(i)=\ulcorner \$ \urcorner)\wedge\\
&\wedge(i>1\wedge i\leq 2^{q(|x|)}\supset\varsigma(i)=\ulcorner b\urcorner)\wedge\\
&\wedge(i>r(x)\dot{-}|x|\dot{-}2\wedge i<r(x)\dot{-}1\wedge 1=Bit(r(x)\dot{-}i\dot{-}2,x)\supset\varsigma(i)=\ulcorner 1\urcorner)\wedge\\
&\wedge(i>r(x)\dot{-}|x|\dot{-}2\wedge i<r(x)\dot{-}1\wedge 0=Bit(r(x)\dot{-}i\dot{-}2,x)\supset\varsigma(i)=\ulcorner 0\urcorner)].
\end{aligned}
$$

Thus $Init_M(\varsigma^n,x)$ asserts that the values of ς^n for $i<r(x)$ code the ID

$$\$\ q_0\ b\ b\ \cdots\ b\ b\ a_{|x|\dot{-}1}\ \cdots\ a_0\ \$$$

where a_j is equal to 0 or 1 depending on the i-th bit of the binary representation of x. (Without loss of generality, we may assume the input to M conforms to the format expressed by $Init_M$.)

We define $Run_M(\varsigma,i,x)$ to mean that ς codes i steps of the running of $M(x)$:

$$
\begin{aligned}
Run_M(\varsigma,i,x) \iff & Init_M(\varsigma,x)\wedge\\
&\wedge(\forall j<i)(\forall k<r(x)\dot{-}2)[Next_M(\varsigma(j\cdot r(x)+k),\varsigma(j\cdot r(x)+k+1),\\
&\qquad \varsigma(j\cdot r(x)+k+2),\varsigma(j\cdot r(x)+k+3),\varsigma((j+1)\cdot r(x)+k+1))]\wedge\\
&\wedge(\forall j\leq i)[\varsigma(j\cdot r(x))=\ulcorner \$ \urcorner\wedge\varsigma((j+1)\cdot r(x)\dot{-}1)=\ulcorner \$ \urcorner].
\end{aligned}
$$

It is easy to see, by use of $\Sigma_0^{1,b}$–FCA, that

$$V_2^1\vdash(\exists\lambda^n)Run_M(\lambda^n,0,x)$$

$$V_2^1\vdash(\exists\lambda^n)Run_M(\lambda^n,a,x)\supset(\exists\lambda^n)Run_M(\lambda^n,Sa,x).$$

Then, by an application of $\Sigma_1^{1,b}$–IND,

$$V_2^1\vdash(\exists\lambda^n)Run_M(\lambda^n,2^{q(|x|)},x).$$

Furthermore, the uniqueness condition is also provable, so

$$V_2^1 \vdash Run_M(\varsigma^n,i,x) \wedge Run_M(\theta^n,i,x) \supset (\forall y<(i+1)\cdot r(x))(\varsigma^n(y)=\theta^n(y)).$$

We can easily $\Sigma_1^{1,b}$–define the functional $Value_M$ such that if ς^n satisfies $Run_M(\varsigma^n,2^{q(|x|)},x)$, then $Value_M(\varsigma^n,x)$ is equal to the output of M which is coded in the last ID coded by ς^n. $Value_M$ is in fact polynomial time (relative to a function oracle for ς^n) and can be $\Sigma_1^b(\varsigma)$–defined.

We are now ready to give the desired formula $A_M(x,y)$ which defines the function f computed by M. The formula $A_M(x,y)$ is defined by

$$A_M(x,y) \iff (\exists\lambda^n)(Run_M(\lambda^n,2^{q(|x|)},x) \wedge y=Value_M(\lambda^n,x)).$$

Because $Value_M$ is polynomial time, we can assume without loss of generality that there is a term $t_M(x)$ such that V_2^1 proves that $(\forall\lambda^n)(Value_M(\lambda^n,x) \le t_M(x))$. Then,

$$V_2^1 \vdash (\forall x)(\exists y \le t_M) A_M(x,y).$$

We can now $\Sigma_1^{1,b}$–define f with the defining axiom

$$f(x)=y \iff A_M(x,y).$$

Q.E.D. □

10.2. PSPACE functions are $\Sigma_1^{1,b}$-definable in U_2^1.

Definition: PSPACE is the set of functions f of polynomial growth rate which can be computed by a Turing machine M_f such that there is a polynomial $p(\vec{n})$ so that the total number of tape squares used by M_f on input $\vec{x}$ is always less than $p(|\vec{x}|)$.

Definition: Let $\phi_1, \ldots, \phi_r$ be predicate variables of a second order theory of Bounded Arithmetic where each ϕ_i is k_i-ary. Then PSPACE$(\phi_1, \ldots, \phi_r)$ is the uniform set of functionals f such that the following hold:

(1′)-(3′): Conditions (1)-(3) of the definition of EXPTIME$(\phi_1, \ldots, \phi_r)$ hold, and

(4′) For some polynomial $p(\vec{n})$, the total tape space used by $M_f(\vec{x},\Omega_1, \ldots, \Omega_r)$ is less than $p(|\vec{x}|)$ for all $\vec{x}$ and all $\Omega_1, \ldots, \Omega_r$.

PSPACE$(\omega_{k_1}^1, \ldots, \omega_{k_r}^1)$ is another name for PSPACE$(\phi_1, \ldots, \phi_r)$.

There is no condition (5′) in the definition above since condition (4′) implies the condition (5) of the definition of EXPTIME($\vec{\phi}$).

Before proving the assertion made by the title of this section we will give an illuminating example. Recall that Theorem 2.7 showed that length bounded counting is Σ_1^b–definable in S_2^1. A more general concept is that of *bounded counting*: a function f is defined by bounded counting from A if $f(y)=(\#z\leq y)A(z)$. Clearly, if A is a PSPACE predicate, then f is a PSPACE function and thus bounded counting should be definable in U_2^1.

We shall use the following scheme to express bounded counting: θ will be a function variable satisfying

$$\theta(x,y) = \begin{cases} \theta(2x,y)+\theta(2x+1,y) & \text{if } x<2^{|y|} \\ 1 & \text{if } A(x\dot{-}2^{|y|}) \text{ and } 2^{|y|}\leq x\leq 2^{|y|}+y \\ 0 & \text{otherwise} \end{cases}$$

Then $\theta(1,y)$ is equal to the number of $x\leq y$ such that $A(x)$ holds.

Proposition 2: Let A be a $\Sigma_0^{1,b}$–formula and let $t(x)$ be any term. Then the function

$$f(y) = (\#z\leq t(y))A(y,z)$$

is $\Sigma_1^{1,b}$–definable in U_2^1.

Proof: First we define $RDEF(\varsigma,x,y)$ to be the formula asserting that $\varsigma(x,y)$ satisfies a condition similar to the definition of θ above; namely, $RDEF(\varsigma,x,y)$ is

$$\begin{aligned}(x<2^{|t|}\supset\varsigma(x,y)=\varsigma(2x,y)+\varsigma(2x+1,y)\wedge|\varsigma(x,y)\dot{-}1|\leq 1+|t|\dot{-}|x|)\wedge\\ \wedge(x\geq 2^{|t|}\wedge x\leq 2^{|t|}+t\supset\varsigma(x,y)\leq 1\wedge(A(x\dot{-}2^{|t|})\leftrightarrow\varsigma(x,y)=1))\wedge\\ \wedge(x>2^{|t|}+t\vee x=0\supset\varsigma(x,y)=0).\end{aligned}$$

Define $B(i,x)$ to be the formula

$$(\exists\varsigma^{2^{|t|}})(\forall z\leq 2^{|t|+1})[|z|\geq(1+|t|)\dot{-}i\supset RDEF(\varsigma^{2^{|t|}},z,x)].$$

An easy application of $\Sigma_0^{1,b}$–*FCR* shows that $U_2^1\vdash B(0,x)$. Similarly, $U_2^1\vdash B(i,x)\supset B(Si,x)$. By $\Sigma_1^{1,b}$–*LIND* ,

$$U_2^1\vdash B(|t|+1,x).$$

Thus,

$$S_2^1 \vdash (\forall y)(\exists v \le 2^{|t|})(\exists \varsigma^{2^{|t|}})\,[v = \varsigma^{2^{|t|}}(1,y) \wedge (\forall x < 2^{|t|+1}) RDEF(\varsigma^{2^{|t|}}, x, y)]\,.$$

This partially proves Proposition 2. We leave it for the reader to show that U_2^1 can prove that the y is unique, and that the bound $2^{|t|}$ on y can be sharpened to t.

Q.E.D. □

The general idea of the proof of Proposition 2 is a "divide and conquer" strategy. In order to compute $\varsigma(1)$, the problem is divided into the two subproblems of computing $\varsigma(2)$ and $\varsigma(3)$. These subproblems are further divided into subproblems, etc. Thus to find $(\#y \le t)A(y)$ we first find $(\#y < 2^{|t| \dot{-} 1})A(y)$ and $(\#y \le t \dot{-} 2^{|t| \dot{-} 1})A(y + 2^{|t| \dot{-} 1})$ and compute the sum. This divide and conquer strategy can be generalized to the concept of *limited recursion.*

Definition: Let g and h be functions with polynomial growth rate and let p and q be suitable polynomials. We say that f is *defined by limited recursion from g and h with time bound p and space bound q* iff the following holds. Let f^* be defined inductively by

$$f^*(\vec{x},y) = \begin{cases} 0 & \text{if } |y| > p(|\vec{x}|) \vee y = 0 \\ g(\vec{x},y) & \text{if } |y| = p(|\vec{x}|) \\ h(\vec{x},y,f^*(\vec{x},2y),f^*(\vec{x},2y+1)) & \text{otherwise} \end{cases}$$

Then, for all y and $\vec{x}$, we must have $|f^*(\vec{x},y)| \le q(|\vec{x}|)$ and f must satisfy the defining equation

$$f(\vec{x}) = f^*(\vec{x},1).$$

The definition of *limited recursion* is somewhat similar to that of *limited iteration,* however, the two concepts are substantially different. The time bound p of limited recursion does not correspond to the runtime of a conventional Turing machine. Instead, p is a measure of the maximum depth of recursion. It will be seen that limited recursion is similar to the action of an alternating Turing machine (ATM) and that p is a measure similar to the runtime of an ATM.

The next theorem states that limited recursion is definable in U_2^1.

Theorem 3: Suppose that g and h are $\Sigma_0^{1,b}$–definable in U_2^1 and that p and q are suitable polynomials. Further suppose that f is defined by limited recursion from g and h with bounds p and q. Then f is $\Sigma_1^{1,b}$–definable in U_2^1.

Proof: The proof is similar to the proof of Proposition 2. We first define $RDEF2(\varsigma,\vec{x},y)$ to be the formula

$$[|y|>p(|\vec{x}|)\vee y=0\supset\varsigma(\vec{x},y)=0]\wedge[|y|=p(|\vec{x}|)\supset\varsigma(\vec{x},y)=\min(g(\vec{x},y),2^{q(|\vec{x}|)})]\wedge$$
$$\wedge[|y|<p(|\vec{x}|)\supset\varsigma(\vec{x},y)=\min(h(\vec{x},y,\varsigma(\vec{x},2y),\varsigma(\vec{x},2y+1)),2^{q(|\vec{x}|)})].$$

So $(\forall y<2^{p(|\vec{x}|)})RDEF2(\varsigma,\vec{x},y)$ asserts that the ς function is equal to the f^* function of the definition of limited recursion. Note that the min function is used in the definition of *RDEF2* to explicitly prevent the possibility of an overflow; that is to say, the possibility that some value of $f^*(\vec{x},y)$ is too large.

We used g and h as function variables in the definition of *RDEF2*; since g and h are $\Sigma_0^{1,b}$–definable, *RDEF2* is a $\Sigma_0^{1,b}$–formula. Let $s(\vec{a})$ be the term $2^{q(|\vec{a}|)}$ and define $B(i,\vec{x},\varsigma)$ to be the formula

$$(\forall y<2^{p(|x|)+1})[|y|\geq p(|\vec{x}|)\dot{-}i\supset RDEF2(\varsigma,\vec{x},y)].$$

It is easy to see using $\Sigma_0^{1,b}$–comprehension that

$$U_2^1\vdash(\exists\lambda^s)B(0,\vec{x},\lambda^s)$$

and

$$U_2^1\vdash(\exists\lambda^s)B(a,\vec{x},\lambda^s)\supset(\exists\lambda^s)B(a+1,\vec{x},\lambda^s).$$

So by $\Sigma_1^{1,b}$–*PIND* , $U_2^1\vdash(\exists\lambda^s)B(p(|\vec{x}|),\vec{x},\lambda^s)$.

We also need to show that U_2^1 proves that the λ^s is unique; that is, we need to show that

$$U_2^1\vdash B(p(|\vec{x}|),\vec{x},\varsigma^s)\wedge B(p(|\vec{x}|),\vec{x},\theta^s)\supset(\forall y<2^{p(|\vec{x}|)})(\theta^s(\vec{x},y)=\varsigma^s(\vec{x},y)).$$

For this purpose, let $C(i,\vec{x},\varsigma^s,\theta^s)$ be the formula

$$(\forall y<2^{p(|\vec{x}|)})[|y|\geq p(|\vec{x}|)\dot{-}i\supset\theta^s(\vec{x},y)=\varsigma^s(\vec{x},y)]$$

and let $D(\vec{x},\varsigma^s,\theta^s)$ be the formula

$$B(p(|\vec{x}|),\vec{x},\varsigma^s)\wedge B(p(|\vec{x}|),\vec{x},\theta^s).$$

Then it is clear that

$$U_2^1\vdash D(\vec{x},\varsigma^s,\theta^s)\supset C(0,\vec{x},\varsigma^s,\theta^s)$$

and

$$U_2^1\vdash D(\vec{x},\varsigma^s,\theta^s)\wedge C(a,\vec{x},\varsigma^s,\theta^s)\supset C(a+1,\vec{x},\varsigma^s,\theta^s)$$

from which the desired uniqueness condition is obtained by an application of $\Sigma_0^{1,b}$–*PIND*.

Let $A(\vec{x},y)$ be the formula

$$(\exists\lambda^s)[B(p(|\vec{x}|),\vec{x},\lambda^s)\wedge y{=}\lambda^s(\vec{x},1)].$$

So $U_2^i\vdash(\forall\vec{x})(\exists!y{\le}s(\vec{x}))A(\vec{x},y)$. Also, for all $\vec{x}$, $A(\vec{x},f(\vec{x}))$ is true. Since A is a $\Sigma_1^{1,b}$–formula, f is by definition $\Sigma_1^{1,b}$–definable.

Q.E.D. □

We are now ready to prove that all PSPACE functions can be $\Sigma_1^{1,b}$–defined in U_2^1.

Theorem 4: Let f be a function with polynomial growth rate in PSPACE. Then f is $\Sigma_1^{1,b}$–definable in U_2^1.

Proof: Chandra, Kozen and Stockmeyer [4] show that the PSPACE predicates are precisely the predicates which can be recognized by polynomial time alternating Turing machines. This is also true for PSPACE functions with polynomial growth rate: if f is of polynomial growth rate then $f\in$PSPACE iff there is a polynomial time alternating Turing machine (i.e., a transducer) which computes f.

But polynomial time alternating Turing machines are easily defined by limited recursion from polynomial time functions g and h. By Theorem 3.1, g and h are $\Sigma_0^{1,b}$–definable in U_2^1. Theorem 3 thus implies that every PSPACE function of polynomial growth rate can be $\Sigma_1^{1,b}$–defined in U_2^1.

Q.E.D. □

10.3. Deterministic PSPACE Turing machines.

Theorem 4 established that U_2^1 can $\Sigma_1^{1,b}$–define the PSPACE functions; however, the proof of Theorem 4 used Chandra, Kozen and Stockmeyer's [4] representation of PSPACE functions by alternating polynomial time Turing machines. An interesting question is whether U_2^1 can prove directly that any polynomial space bounded, deterministic Turing machine will run to completion.

That is, let M be a PSPACE Turing machine for which there is a term $r(x){=}|t(x)|$ with $r(x)\ge|x|+3$ for all x so that M is constrained by tape markers to run in space $r(x)$ on input x. Let Run_M be defined exactly as in §10.1. Then our question is whether

$$U_2^1\vdash(\exists\lambda^n)Run_M(\lambda^n,2^{q(|x|)},x)$$

where q is any polynomial. The answer to this question is affirmative:

Theorem 5: Let M be a deterministic Turing machine constrained by tape markers to run in space $r(x)=|t(x)|$ on input x, as above. Then

$$U_2^1 \vdash (\forall y)(\exists \lambda^n) Run_M(\lambda^n, y, a).$$

Proof: Let Run_M and $Init_M$ be defined as in §10.1. We need the ability to code a state by an integer, so we introduce the following functional:

$$\begin{aligned} STATE_M(\varsigma^n, i, a)=b \iff & |b| \leq |n| \cdot |t(a)| \wedge \\ & \wedge (\forall v < |t(a)|)(LSP(MSP(b, v \cdot |n|), |n|) = \varsigma^n(i \cdot r(a)+v)). \end{aligned}$$

(Recall that n bounds the Gödel numbers of symbols used to code states.) Thus $STATE_M(\varsigma^n, i, a)$ is equal to a number which codes the i-th state of the run coded by ς^n.

Let $PRun_M$ be the formula

$$\begin{aligned} PRun_M(\varsigma, i, a) \iff & (\forall j < i)(\forall k < r(a) \dot{-} 2)[Next_M(\varsigma(j \cdot r(a)+k), \varsigma(j \cdot r(a)+k+1), \\ & \varsigma(j \cdot r(a)+k+2), \varsigma(j \cdot r(a)+k+3), \varsigma((j+1) \cdot r(a)+k+1))] \wedge \\ & \wedge (\forall j \leq i)[\varsigma(j \cdot r(a)) = \ulcorner \$ \urcorner \wedge \varsigma((j+1) \cdot r(a) \dot{-} 1) = \ulcorner \$ \urcorner]. \end{aligned}$$

So $PRun_M(\varsigma, i, a)$ asserts that ς codes $i+1$ states of a run of M except that no conditions are put on the initial state coded by ς. Compare the definition of $PRun_M$ with the definition of Run_M.

Let $D_M(c, a)$ be the formula

$$\begin{aligned} D_M(c, a) \iff & (\forall x < 2^{|n| \cdot |t(a)|})[LSP(x, |n|) = \ulcorner \$ \urcorner \wedge MSP(x, r(a) \dot{-} |n|) = \ulcorner \$ \urcorner \supset \\ & \supset (\exists \lambda^n)(x = STATE_M(\lambda^n, 0, a) \wedge PRun_M(\lambda^n, c, a))]. \end{aligned}$$

$D_M(c, a)$ asserts that for all possible initial states there is a ς^n which codes $c+1$ states of a run of M beginning with that state. Note that because M is polynomially space bounded, a first order bounded quantifier can be used to quantify over all possible initial states x. It is clear that

$$U_2^1 \vdash D_M(0, a).$$

Also, we claim that

$$U_2^1 \vdash D_M(\lfloor \tfrac{1}{2} c \rfloor, a) \supset D_M(c, a).$$

To prove the claim we argue informally in U_2^1. Suppose $D_M(\lfloor \frac{1}{2} c \rfloor)$ is true and that $x < 2^{|n| \cdot |t(a)|}$ codes a state for M. Then there exists a λ_1^n such that $x = STATE_M(\lambda_1^n, 0, a)$ and such that $PRun_M(\lambda_1^n, \lfloor \frac{1}{2} c \rfloor, a)$. So let $x_2 = STATE_M(\lambda_1^n, \lfloor \frac{1}{2} c \rfloor, a)$ and let λ_2^n be such that

$x_2{=}STATE_M(\lambda_2^n,0,a)$ and such that $PRun_M(\lambda_2^n,\lfloor\frac{1}{2}c\rfloor,a)$. Define λ^n by putting λ_1^n and λ_2^n together so that $x{=}STATE_M(\lambda^n,0,a)$ and $PRun_M(\lambda^n,2{\cdot}\lfloor\frac{1}{2}c\rfloor,a)$. If c is even, we are done. If c is odd we easily add one more state to the end of the run coded by λ^n to get the desired result. This proves the claim.

Since D_M is a $\Sigma_1^{1,b}$-formula, we can use $\Sigma_1^{1,b}$-*PIND* to deduce that

$$U_2^1 \vdash (\forall y) D_M(y,a).$$

From this Theorem 5 follows easily.

Q.E.D. □

10.4. Witnessing a $\Sigma_1^{1,b}$-Formula.

Our next main goal is to prove the converses of Theorems 1 and 4; this will be accomplished by a proof similar to the proof of Theorem 5.5. This section establishes some preliminary definitions and propositions needed for the proofs in §10.5 and §10.6.

For the next three sections, we shall work exclusively in the theories $\mathring{U}_2^i(\delta)$ and $\mathring{V}_2^i(\delta)$.

We define *Witness2* below analogously to the way *Witness* was defined in §5.1. When A is a $\Sigma_1^{1,b}(\delta)$-formula with free first order variables $\vec{a}$ and with free second order variables $\vec{\alpha}$, we define $Witness2_A^{\vec{a},\vec{\alpha}}(\gamma,\vec{a},\vec{\alpha})$ to be a $\Sigma_0^{1,b}(\delta)$-formula which asserts that γ is a predicate which "witnesses" the truth of $A(\vec{a},\vec{\alpha})$.

Although $Witness2_A$ could readily be defined for arbitrary bounded formulae A, we shall forsake the added generality and restrict A to be a $\Sigma_1^{1,b}(\delta)$-formula.

Definition: Suppose A is a $\Sigma_1^{1,b}(\delta)$-formula. Let the free first order variables of A be among $\vec{a}$ and the free second order (predicate) variables of A be among $\vec{\alpha}$. The $\Sigma_0^{1,b}(\delta)$-formula $Witness2_A^{\vec{a},\vec{\alpha}}(\gamma,\vec{a},\vec{\alpha})$ is defined below, where γ is a unary predicate variable. The definition is by induction on the complexity of A.

(1) If A is a $\Sigma_0^{1,b}(\delta)$-formula, then define

$$Witness2_A^{\vec{a},\vec{\alpha}}(\gamma,\vec{a},\vec{\alpha}) \iff A(\vec{a},\vec{\alpha})$$

(2) If A is $B\wedge C$, define

$$Witness2_A^{\vec{a},\vec{\alpha}}(\gamma,\vec{a},\vec{\alpha}) \iff Witness2_B^{\vec{a},\vec{\alpha}}(\boldsymbol{\beta}(1,\gamma),\vec{a},\vec{\alpha})\wedge$$
$$\wedge\, Witness2_C^{\vec{a},\vec{\alpha}}(\boldsymbol{\beta}(2,\gamma),\vec{a},\vec{\alpha})$$

(3) If A is $B \vee C$, define

$$Witness2_A^{\vec{a},\vec{\alpha}}(\gamma,\vec{a},\vec{\alpha}) \iff Witness2_B^{\vec{a},\vec{\alpha}}(\boldsymbol{\beta}(1,\gamma),\vec{a},\vec{\alpha}) \vee \vee\, Witness2_C^{\vec{a},\vec{\alpha}}(\boldsymbol{\beta}(2,\gamma),\vec{a},\vec{\alpha})$$

(4) If A is $(\forall x \le t)B(x)$, then define

$$Witness2_A^{\vec{a},\vec{\alpha}}(\gamma,\vec{a},\vec{\alpha}) \iff (\forall x \le t)\, Witness2_B^{\vec{a},x,\vec{\alpha}}(\boldsymbol{\beta}(x+1,\gamma),\vec{a},x,\vec{\alpha})$$

(5) If A is $(\exists x \le t)B(x)$, then define

$$Witness2_A^{\vec{a},\vec{\alpha}}(\gamma,\vec{a},\vec{\alpha}) \iff (\exists x \le t)\, Witness2_B^{\vec{a},x,\vec{\alpha}}(\gamma,\vec{a},x,\vec{\alpha})$$

(6) If A is $(\exists \phi^k)B(\phi^k)$ where ϕ^k is a k-ary predicate variable, then define

$$Witness2_A^{\vec{a},\vec{\alpha}}(\gamma,\vec{a},\vec{\alpha}) \iff Witness2_B^{\vec{a},\vec{\alpha},\phi^k}(\boldsymbol{\beta}(2,\gamma),\vec{a},\vec{\alpha},ARY_k(\boldsymbol{\beta}(1,\gamma)))$$

(7) If A is $\neg B$ and $A \notin \Sigma_0^{1,b}(\delta)$, then define $Witness2_A^{\vec{a},\vec{\alpha}}$by using prenex operations to transform A so that it can be handled by Cases (1)-(6). Specifically, if A is $\neg(\neg B)$, $\neg(B \wedge C)$, $\neg(B \vee C)$, $\neg(\forall x \le t)B$, $\neg(\exists x \le t)B$ or $\neg(\forall \phi)B$; let A^* be B, $(\neg B)\vee(\neg C)$, $(\neg B)\wedge(\neg C)$, $(\exists x \le t)(\neg B)$, $(\forall x \le t)(\neg B)$ or $(\exists \phi)(\neg B)$. Then define

$$Witness2_A^{\vec{a},\vec{\alpha}}(\gamma,\vec{a},\vec{\alpha}) \iff Witness2_{A^*}^{\vec{a},\vec{\alpha}}(\gamma,\vec{a},\vec{\alpha}).$$

Proposition 6: Let $A(\vec{a},\vec{\alpha})$ be any $\Sigma_1^{1,b}(\delta)$–formula. Then $\mathring{U}_2^1(\delta)$ and $\mathring{V}_2^1(\delta)$ prove

$$A(\vec{a},\vec{\alpha}) \leftrightarrow (\exists \psi)\, Witness2_A^{\vec{a},\vec{\alpha}}(\psi,\vec{a},\vec{\alpha}).$$

Proof: by induction on the complexity of A. The only nontrivial cases are (4) and (6) in the definition of *Witness2*.

Case (4): Suppose A is $(\forall x \le t)B(x)$. The induction hypothesis is that $\mathring{U}_2^1(\delta)$ and $\mathring{V}_2^1(\delta)$ prove

$$B(x,\vec{a},\vec{\alpha}) \leftrightarrow (\exists \psi)\, Witness2_B^{x,\vec{a},\vec{\alpha}}(\psi,x,\vec{a},\vec{\alpha}).$$

By $\Sigma_1^{1,b}$–replacement (Theorem 9.16), $\mathring{U}_2^1(\delta)$ and $\mathring{V}_2^1(\delta)$ prove

$$(\forall x \le t)(\exists \psi)\, Witness2_B^{x,\vec{a},\vec{\alpha}}(\psi,x,\vec{a},\vec{\alpha}) \leftrightarrow$$
$$\leftrightarrow (\exists \psi)(\forall x \le t)\, Witness2_B^{x,\vec{a},\vec{\alpha}}(\boldsymbol{\beta}(x+1,\psi),x,\vec{a},\vec{\alpha})$$

from which the desired result is immediate.

Case (6): Suppose A is $(\exists\phi)B(\phi)$ and that $\mathring{U}_2^1(\delta)$ and $\mathring{V}_2^1(\delta)$ prove

$$B(\vec{a},\vec{\alpha},\phi) \leftrightarrow (\exists \psi)\, Witness2_B^{\vec{a},\vec{\alpha},\phi}(\psi,\vec{a},\vec{\alpha},\phi)$$

where ϕ is a k-ary predicate variable. From the definition of $\boldsymbol{\beta}$ and ARY_k, we have immediately that $\mathring{U}_2^1(\delta)$ and $\mathring{V}_2^1(\delta)$ prove

$$(\exists\phi)(\exists \psi)\, Witness2_B^{\vec{a},\vec{\alpha},\phi}(\psi,\vec{a},\vec{\alpha},\phi) \leftrightarrow$$
$$\leftrightarrow (\exists \psi)\, Witness2_B^{\vec{a},\vec{\alpha},\phi}(\boldsymbol{\beta}(2,\psi),\vec{a},\vec{\alpha},ARY_k(\boldsymbol{\beta}(1,\psi)))$$

and from this the desired result is immediate.

Q.E.D. □

As we remarked above, $Witness2_A^{\vec{a},\vec{\alpha}}$ is a $\Sigma_0^{1,b}(\delta)$–formula whenever A is a $\Sigma_1^{1,b}(\delta)$–formula. The next proposition specifies the computational complexity of $Witness2_A^{\vec{a},\vec{\alpha}}$.

Proposition 7: Let $A(\vec{a},\vec{\alpha})$ be a $\Sigma_1^{1,b}(\delta)$–formula. Then $Witness2_A^{\vec{a},\vec{\alpha}}(\gamma,\vec{a},\vec{\alpha})$ represents a predicate in PSPACE($\gamma,\vec{\alpha}$).

Proof: This is an immediate consequence of the fact that $Witness2_A^{\vec{a},\vec{\alpha}}$ contains no second order quantifiers. □

Lemma 8: Let $A(\vec{a},\vec{\alpha},\beta)$ be a $\Sigma_1^{1,b}(\delta)$–formula and let $B(\vec{c},\vec{a},\vec{\alpha})$ be a $\Sigma_0^{1,b}(\delta)$–formula, where the free variables of A and B are as indicated. Furthermore, β is a k-ary predicate variable and $\vec{c}$ is a vector of k first order variables. Let U be the abstract $\{\vec{x}\}B(\vec{x},\vec{a},\vec{\alpha})$ and let $A^*(\vec{a},\vec{\alpha})$ be the formula $A(\vec{a},\vec{\alpha},U)$. Then $\mathring{U}_2^1(\delta)$ and $\mathring{V}_2^1(\delta)$ prove

$$Witness2_A^{\vec{a},\vec{\alpha},\beta}(\gamma,\vec{a},\vec{\alpha},U) \leftrightarrow Witness2_{A^*}^{\vec{a},\vec{\alpha}}(\gamma,\vec{a},\vec{\alpha}).$$

Proof: This is easily proved by induction on the complexity of A. □

The final lemma of this section is not directly concerned with the *Witness2* metaformula, but it will be useful in the proofs of the theorems of §10.5 and §10.6. Intuitively, it states that if $A(\alpha)$ is a bounded formula then the truth value of $A(\alpha)$ does not depend on all of α's values but only on α restricted to some bounded domain.

Lemma 9: Let $A(\alpha,\vec{a},\vec{\gamma})$ be a bounded formula with all free variables as indicated. For notational simplicity, further suppose α is a unary predicate variable. Then there is a term $s_A(\vec{a})$ such that $\mathring{U}_2^1(\delta)$ and $\mathring{V}_2^1(\delta)$ prove

$$(\forall z \le s_A(\vec{a}))(\alpha(z) \leftrightarrow \beta(z)) \supset [A(\alpha,\vec{a},\vec{\gamma}) \leftrightarrow A(\beta,\vec{a},\vec{\gamma})].$$

Proof: This is readily proved by induction on the complexity of A. $\square$

As in Chapter 5, we adopt the convention that conjunction and disjunction associate from right to left. We also extend our use of the $\ll \cdots \gg$ notation to apply to predicates. So

$$\ll \alpha_1, \ldots, \alpha_n \gg$$

denotes $< \alpha_1, < \alpha_2, \ldots, < \alpha_{n-1}, \alpha_n > > >$.

10.5. Only PSPACE is $\Sigma_1^{1,b}$-definable in U_2^1.

In this section, the converse to Theorem 4 is proved. This establishes that a function f of polynomial growth rate is $\Sigma_1^{1,b}$–definable in U_2^1 iff f is computed by some polynomial space bounded (PSPACE) Turing machine. The main theorem of this section is:

Theorem 10: Suppose $A(\vec{c},d)$ is a $\Sigma_1^{1,b}(\delta)$–formula where $\vec{c}$ and d are all the free variables of A. Also suppose $\mathring{U}_2^1(\delta) \vdash (\forall \vec{x})(\exists y)A(\vec{x},y)$. Then there is a $\Delta_1^{1,b}(\delta)$–formula B, a term t and a function f so that

(1) $\mathring{U}_2^1(\delta) \vdash (\forall \vec{x})(\forall y)(B(\vec{x},y) \supset A(\vec{x},y))$

(2) $\mathring{U}_2^1(\delta) \vdash (\forall \vec{x})(\exists y \le t)B(\vec{x},y)$

(3) $\mathring{U}_2^1(\delta) \vdash (\forall \vec{x})(\forall y)(\forall z)(B(\vec{x},y) \wedge B(\vec{x},z) \supset y=z)$

(4) For all $\vec{n}$, $\mathbf{N} \models B(\vec{n},f(\vec{n}))$

(5) f is a PSPACE function

Hence, f is a PSPACE function which is $\Sigma_1^{1,b}(\delta)$–definable in $\mathring{U}_2^1(\delta)$ and for all $\vec{n}$, $A(\vec{n},f(\vec{n}))$ is true.

The converse of Theorem 4 is an immediate corollary of Theorem 10:

Corollary 11: Suppose $A(\vec{c},d)$ is a $\Sigma_1^{1,b}$–formula where $\vec{c}$ and d are all the free variables of A. Also suppose $U_2^1 \vdash (\forall\vec{x})(\exists y)A(\vec{x},y)$. Then there is a PSPACE function f such that for all $\vec{n}$, $\mathbf{N} \models A(\vec{n},f(\vec{n}))$.

Proof: of Corollary 11 from Theorem 10:

By Lemma 9.6 and Theorem 9.5, we can assume without loss of generality that $A \in \tilde{\Sigma}_1^{1,b}$ and that $\tilde{U}_2^1 \vdash (\forall\vec{x})(\exists y)A(\vec{x},y)$. But $\mathring{U}_2^1(\delta)$ is an extension of $\tilde{U}_2^1$, so $\mathring{U}_2^1(\delta) \vdash (\forall\vec{x})(\exists y)A(\vec{x},y)$ and Theorem 10 states that the desired function f exists. □

We shall prove Theorem 10 by proving a more general theorem:

Theorem 12: Suppose $\mathring{U}_2^1(\delta) \vdash \Gamma,\Pi \longrightarrow \Lambda,\Delta$ and each formula in $\Gamma\cup\Delta$ is a $\Sigma_1^{1,b}(\delta)$–formula and each formula in $\Pi\cup\Lambda$ is a $\Pi_1^{1,b}(\delta)$–formula. Let $c_1,\ldots,c_p$ and $\gamma_1,\ldots,\gamma_q$ be the free variables in $\Gamma,\Pi \longrightarrow \Lambda,\Delta$. Let X and Y be the $\Sigma_1^{1,b}$–formulae

$$X = (\bigwedge\Gamma)\wedge\bigwedge\{\neg C : C\in\Lambda\}$$

and

$$Y = (\bigvee\Delta)\vee\bigvee\{\neg C : C\in\Pi\}.$$

Then there is a PSPACE$(\alpha,\vec{\gamma})$ predicate M so that

(1) M is $\Delta_1^{1,b}(\delta)$–defined by $\mathring{U}_2^1(\delta)$ and

(2) $\mathring{U}_2^1(\delta) \vdash \mathit{Witness2}_X^{\vec{c},\vec{\gamma}}(\alpha,\vec{c},\vec{\gamma}) \supset \mathit{Witness2}_Y^{\vec{c},\vec{\gamma}}(\{x\}M(x,\vec{c},\alpha,\vec{\gamma}),\vec{c},\vec{\gamma})$.

Proof: of Theorem 10 from Theorem 12:

The hypothesis of Theorem 10 is that $\mathring{U}_2^1(\delta) \vdash (\forall\vec{x})(\exists y)A(\vec{x},y)$. By the extension of Parikh's theorem to second order Bounded Arithmetic, there is a term t such that $\mathring{U}_2^1(\delta) \vdash (\forall\vec{x})(\exists y \le t(\vec{x}))A(\vec{x},y)$. We now apply Theorem 12 with $\Delta = \{(\exists y \le t(\vec{c}))A(\vec{c},y)\}$ and with $\Gamma=\Pi=\Lambda=\emptyset$. Theorem 12 asserts that there is a PSPACE predicate M which is $\Delta_1^{1,b}(\delta)$–defined by $\mathring{U}_2^1(\delta)$ so that

$$\mathring{U}_2^1(\delta) \vdash \mathit{Witness2}_{\bigvee\Delta}^{\vec{c}}(\{x\}M(x,\vec{c}),\vec{c}).$$

By the definition of *Witness2*, this means that

$$\mathring{U}_2^1(\delta) \vdash (\exists y \le t(\vec{c}))\,\mathit{Witness2}_{A(\vec{c},d)}^{\vec{c},d}(\{x\}M(x,\vec{c}),\vec{c},y).$$

Now define

$$f(\vec{c}) \;=\; (\mu y)\, Witness2^{\vec{c},d}_{A(\vec{c},d)}(\{x\}M(x,\vec{c}),\vec{c},y)$$

and

$$B(\vec{c},d) \iff Witness2^{\vec{c},d}_{A(\vec{c},d)}(\{x\}M(x,\vec{c}),\vec{c},d)\wedge$$
$$\wedge(\forall y<d)(\neg\, Witness2^{\vec{c},d}_{A(\vec{c},d)}(\{x\}M(x,\vec{c}),\vec{c},y)).$$

Since $Witness2^{\vec{c},d}_{A(\vec{c},d)}(\alpha,\vec{c},d)$ is a PSPACE(α) predicate and since $\{x\}M(x,\vec{c})$ is a PSPACE predicate, f is readily seen to be polynomial space computable. Also, since $Witness2^{\vec{c},d}_{A(\vec{c},d)}$ is a $\Sigma_0^{1,b}$–formula and M is a $\Delta_1^{1,b}(\delta)$–defined predicate, B is a $\Delta_1^{1,b}(\delta)$–formula.

It now follows from Theorem 9.15(b) that conditions (1)-(5) of Theorem 10 hold.

Q.E.D. □

Theorem 9.13 showed that an inductive definition similar to but stronger than limited recursion could be defined in $\mathring{U}_2^1(\delta)$. Before we can prove Theorem 10, we need a lemma about the computational complexity of the inductive definition of Theorem 9.13.

Lemma 13: Let $A(a,\vec{c},\vec{\gamma})$ and $B(a,b,\vec{c},\alpha,\vec{\gamma})$ be $\Delta_1^{1,b}(\delta)$–formulae of $\mathring{U}_2^1(\delta)$ where α is a unary predicate variable. Let $t(b,\vec{c})$ be a term. Let $K(a,b,\vec{c},\vec{\gamma})$ be defined from A and B as in Theorem 9.13 by

$$K(a,b,\vec{c},\vec{\gamma}) \iff \begin{cases} A(a,\vec{c},\vec{\gamma}) & \text{if } b{=}0 \text{ and } a\le t(b,\vec{c}) \\ 0{=}1 & \text{if } a>t(b,\vec{c}) \\ B(a,b,\vec{c},\{x\}K(x,\lfloor\frac{1}{2}b\rfloor,\vec{c},\vec{\gamma}),\vec{\gamma}) & \text{otherwise} \end{cases}$$

Then $K(a,b,\vec{c},\vec{\gamma})$ is $\Delta_1^{1,b}(\delta)$–definable by $\mathring{U}_2^1(\delta)$. Furthermore, if A is in PSPACE($\vec{\gamma}$) and B is in PSPACE($\alpha,\vec{\gamma}$) then K is a PSPACE($\vec{\gamma}$) predicate.

Proof: The fact that $K(a,b,\vec{c},\vec{\gamma})$ is $\Delta_1^{1,b}(\delta)$–defined by $\mathring{U}_2^1(\delta)$ is proved by the proof of Theorem 9.13. So we must prove K is in PSPACE($\vec{\gamma}$). To do this we specify an algorithm to compute $K(a,b,\vec{c},\vec{\gamma})$.

Suppose $b\neq 0$ and $a\le t(b,\vec{c})$, then to compute $K(a,b,\vec{c},\vec{\gamma})$ we begin by computing $B(a,b,\vec{c},\alpha,\vec{\gamma})$ with a PSPACE machine M_B with oracles for α and $\vec{\gamma}$. However, we modify M_B so that whenever M_B would have queried the oracle of $\alpha(x)$, instead M_B saves its current state and begins to compute $K(x,\lfloor\frac{1}{2}b\rfloor,\vec{c},\vec{\gamma})$. This process iterates until we wish to compute $K(x,0,\vec{c},\vec{\gamma})$ for some x. Then we just compute $A(x,\vec{c},\vec{\gamma})$ and return its value.

It is straightforward to verify that this algorithm uses only polynomial space.

Q.E.D. □

Proof: of Theorem 12:

By Theorem 9.20 there is a $\mathring{U}_2^1(\delta)$–proof P of $\Gamma,\Pi \longrightarrow \Lambda,\Delta$ such that P is free cut free and in free variable normal form. Hence, by Corollary 9.21, every formula in P is in $\Sigma_1^{1,b}(\delta) \cup \Pi_1^{1,b}(\delta)$.

The proof P will generally contain a number of relational symbols $\delta_1, \ldots, \delta_i$. These relationals are introduced with defining equations $\delta_j(\vec{s},\vec{\alpha}) \leftrightarrow A_j(\vec{s},\vec{\alpha})$ where $A_j \in \Sigma_1^{1,b}(\delta)$. Thus the proof P requires auxiliary proofs $P_1, \ldots, P_i$ of equivalences $A_j(\vec{b},\vec{\beta}) \leftrightarrow B_j(\vec{b},\vec{\beta})$ where each B_j is a $\Pi_1^{1,b}(\delta)$–formula. These auxiliary proofs may themselves use further relational symbols and require their own auxiliary proofs. However, eventually this process must stop and there are proofs $P_1, \ldots, P_k$ such that for every relational symbol δ_j appearing in any of $P, P_1, \ldots, P_k$ which is defined by $\delta_j(\vec{s},\vec{\alpha}) \leftrightarrow A_j(\vec{s},\vec{\alpha})$ there is a $\Pi_1^{1,b}(\delta)$–formula B_j and there are two proofs among $P_1, \ldots, P_k$ of $A_j(\vec{b},\vec{\beta}) \longrightarrow B_j(\vec{b},\vec{\beta})$ and $B_j(\vec{b},\vec{\beta}) \longrightarrow A_j(\vec{b},\vec{\beta})$. In addition, we may assume that each proof $P, P_1, \ldots, P_k$ is free cut free and that every formula appearing in $P, P_1, \ldots, P_k$ is in $\Sigma_1^{1,b}(\delta) \cup \Pi_1^{1,b}(\delta)$.

The proof of Theorem 12 is by induction on the total number of sequents in the proofs $P, P_1, \ldots, P_k$. The argument splits into cases depending on the final inference of P.

First consider the case where P has no inferences and P consists of a single initial sequent. The only difficult case is where P is a defining axiom for a relational, say P is the initial sequent

$$\delta_j(\vec{s},\vec{\gamma}) \longrightarrow A_j(\vec{s},\vec{\gamma})$$

where $\Gamma = \{\delta_j(\vec{s},\vec{\gamma})\}$ and $\Delta = \{A_j(\vec{s},\vec{\gamma})\}$. Then by assumption there is a proof P_j of

$$B_j(\vec{b},\vec{\beta}) \longrightarrow A_j(\vec{b},\vec{\beta})$$

where $B_j \in \Pi_1^{1,b}(\delta)$. By the induction hypothesis, applied to P_j, there is a PSPACE($\vec{\beta}$) predicate G which is $\Delta_1^{1,b}(\delta)$–defined by $\mathring{U}_2^1(\delta)$ such that

$$\mathring{U}_2^1(\delta) \vdash \mathit{Witness2}_{A_j}^{\vec{b},\vec{\beta}}(\boldsymbol{\beta}(1,\{x\}G(x,\vec{b},\vec{\beta})),\vec{b},\vec{\beta}) \vee$$
$$\vee\, \mathit{Witness2}_{\neg B_j}^{\vec{b},\vec{\beta}}(\boldsymbol{\beta}(2,\{x\}G(x,\vec{b},\vec{\beta})),\vec{b},\vec{\beta}).$$

Since

$$\mathring{U}_2^1(\delta) \vdash \mathit{Witness2}_{\neg B_j}^{\vec{b},\vec{\beta}}(\alpha,\vec{b},\vec{\beta}) \supset \neg B_j(\vec{b},\vec{\beta})$$

and $\mathring{U}_2^1(\delta) \vdash \neg B_j \supset \neg A_j$, we have

$$\mathcal{U}_2^1(\delta) \vdash \delta_j(\vec{b},\vec{\gamma}) \supset Witness2_{A_j}^{\vec{b},\vec{\beta}}(\boldsymbol{\beta}(1,\{x\}G(x,\vec{b},\vec{\beta})),\vec{b},\vec{\beta}).$$

So set M to be the PSPACE($\vec{\gamma}$) predicate defined by $M(x,\vec{b},\alpha,\vec{\beta}) \Longleftrightarrow G(<1,x>,\vec{b},\vec{\beta})$. Now since $\delta_j(\vec{s},\vec{\gamma})$ is atomic, we have

$$\mathcal{U}_2^1(\delta) \vdash Witness2_{\delta_j(\vec{b},\vec{\beta})}^{\vec{b},\vec{\beta}}(\alpha,\vec{s},\vec{\gamma}) \supset Witness2_{A_j(\vec{b},\vec{\beta})}^{\vec{b},\vec{\beta}}(\{x\}M(x,\vec{s},\alpha,\vec{\gamma}),\vec{s},\vec{\gamma}).$$

This proves the theorem for the case where P is a single initial sequent of the form $\delta_j(\vec{s},\vec{\gamma}) \longrightarrow A_j(\vec{s},\vec{\gamma})$. The other cases for P a single sequent are similar or easier.

Note that the argument above shows that, no matter how many inferences are in P, every relational symbol $\delta_j(\vec{s},\vec{\gamma})$ appearing in P is a PSPACE($\vec{\gamma}$) predicate.

Next we consider the case where P does contain one or more inferences. We shall henceforth make the simplifying assumption that Π and Λ are the empty cedent. As in the proof of Theorem 5.5 this involves no loss of generality since ($\neg$:left) and ($\neg$:right) inferences can be used to move formulae from side to side and since each inference has a dual. The argument splits into 16 cases depending on the last inference of P.

We shall number the cases as in the proof of Theorem 5.5. We shall omit many of the cases since the argument parallels that of Theorem 5.5 very closely.

Cases (1)-(2): Omitted.

Case (3): ($\vee$:left). Suppose the last inference of P is

$$\frac{B,\Gamma^* \longrightarrow \Delta \qquad\qquad C,\Gamma^* \longrightarrow \Delta}{B \vee C,\Gamma^* \longrightarrow \Delta}$$

Let D be the formula $B \wedge (\bigwedge \Gamma^*)$, let E be $C \wedge (\bigwedge \Gamma^*)$ and let F be $(B \vee C) \wedge (\bigwedge \Gamma^*)$. The induction hypothesis is that there are PSPACE($\vec{\gamma}$) predicates G and H which are $\Delta_1^{1,b}(\delta)$–defined by $\mathcal{U}_2^1(\delta)$ such that

$$\mathcal{U}_2^1(\delta) \vdash Witness2_D^{\vec{c},\vec{\gamma}}(\alpha,\vec{c},\vec{\gamma}) \supset Witness2_{\bigvee\Delta}^{\vec{c},\vec{\gamma}}(\{x\}G(x,\vec{c},\alpha,\vec{\gamma}),\vec{c},\vec{\gamma})$$

$$\mathcal{U}_2^1(\delta) \vdash Witness2_E^{\vec{c},\vec{\gamma}}(\alpha,\vec{c},\vec{\gamma}) \supset Witness2_{\bigvee\Delta}^{\vec{c},\vec{\gamma}}(\{x\}H(x,\vec{c},\alpha,\vec{\gamma}),\vec{c},\vec{\gamma}).$$

Define M by

$$M(x,\vec{c},\alpha,\vec{\gamma}) \iff \begin{cases} G(x,\vec{c},<\boldsymbol{\beta}(1,\boldsymbol{\beta}(1,\alpha)),\boldsymbol{\beta}(2,\alpha)>,\vec{\gamma}) \\ \qquad\qquad \text{if } \ \mathit{Witness2}_B^{\vec{c},\vec{\gamma}}(\boldsymbol{\beta}(1,\boldsymbol{\beta}(1,\alpha)),\vec{c},\vec{\gamma}) \\ H(x,\vec{c},<\boldsymbol{\beta}(2,\boldsymbol{\beta}(1,\alpha)),\boldsymbol{\beta}(2,\alpha)>,\vec{\gamma}) \\ \qquad\qquad \text{otherwise} \end{cases}$$

Clearly M is a PSPACE$(\alpha,\vec{\gamma})$ predicate and is $\Delta_1^{1,b}(\delta)$–definable by $\mathring{U}_2^1(\delta)$ since G, H and $\mathit{Witness2}_B^{\vec{c},\vec{\gamma}}$ are. It is now easy to see that

$$\mathring{U}_2^1(\delta) \vdash \mathit{Witness2}_F^{\vec{c},\vec{\gamma}}(\alpha,\vec{c},\vec{\gamma}) \supset \mathit{Witness2}_{\bigvee\Delta}^{\vec{c},\vec{\gamma}}(\{x\}M(x,\vec{c},\alpha,\vec{\gamma}),\vec{c},\vec{\gamma}).$$

Cases (4)-(13): Omitted.

Case (14): (second order $\exists$:left). Suppose the last inference of P is

$$\frac{B(\beta),\Gamma^* \longrightarrow \Delta}{(\exists\phi)B(\phi),\Gamma^* \longrightarrow \Delta}$$

where β and ϕ are k-ary predicate variables and β is the eigenvariable and must not appear in the lower sequent.

Let D be the formula $B(\beta)\wedge(\bigwedge\Gamma^*)$ and let E be $(\exists\phi)B(\phi)\wedge(\bigwedge\Gamma^*)$. The induction hypothesis is that there is a PSPACE$(\alpha,\beta,\vec{\gamma})$ predicate G which is $\Delta_1^{1,b}(\delta)$–defined by $\mathring{U}_2^1(\delta)$ such that

$$\mathring{U}_2^1(\delta) \vdash \mathit{Witness2}_D^{\vec{c},\beta,\vec{\gamma}}(\alpha,\vec{c},\beta,\vec{\gamma}) \supset \mathit{Witness2}_{\bigvee\Delta}^{\vec{c},\vec{\gamma}}(\{x\}G(x,\vec{c},\alpha,\beta,\vec{\gamma}),\vec{c},\vec{\gamma}).$$

Note we can omit β from the superscript on the lefthand side of this implication since β does not appear in Δ.

Let M be the predicate $\Delta_1^{1,b}(\delta)$–defined by

$$M(x,\vec{c},\alpha,\vec{\gamma}) \iff G(x,\vec{c},\boldsymbol{\beta}(2,\alpha),ARY_k(\boldsymbol{\beta}(1,\alpha)),\vec{\gamma}).$$

Clearly, M is in PSPACE$(\alpha,\vec{\gamma})$ since G is in PSPACE$(\alpha,\beta,\vec{\gamma})$. Furthermore it is easy to see that

$$\mathring{U}_2^1(\delta) \vdash \mathit{Witness2}_E^{\vec{c},\vec{\gamma}}(\alpha,\vec{c},\vec{\gamma}) \supset \mathit{Witness2}_{\bigvee\Delta}^{\vec{c},\vec{\gamma}}(\{x\}M(x,\vec{c},\alpha,\vec{\gamma}),\vec{c},\vec{\gamma}).$$

Case (15): (second order $\exists$:right). Suppose the last inference of P is

$$\frac{\Gamma \longrightarrow B(V),\Delta^*}{\Gamma \longrightarrow (\exists\phi)B(\phi),\Delta^*}$$

where ϕ is a k-ary predicate variable and V is the abstract $\{y_1,\ldots,y_k\}A(y_1,\ldots,y_k,\vec{c},\vec{\gamma})$ where A is a $\Sigma_0^{1,b}(\delta)$–formula.

Let D be the formula $B(V)\vee(\bigvee\Delta^*)$ and let E be $(\exists\phi)B(\phi)\vee(\bigvee\Delta^*)$. The induction hypothesis is that there is a PSPACE($\alpha,\vec{\gamma}$) predicate G which is $\Delta_1^{1,b}(\delta)$–defined by $\mathring{U}_2^1(\delta)$ such that

$$\mathring{U}_2^1(\delta)\vdash \mathit{Witness2}_{\bigwedge\Gamma}^{\vec{c},\vec{\gamma}}(\alpha,\vec{c},\vec{\gamma})\supset \mathit{Witness2}_D^{\vec{c},\vec{\gamma}}(\{x\}G(x,\vec{c},\alpha,\vec{\gamma}),\vec{c},\vec{\gamma}).$$

Let M be the predicate $\Delta_1^{1,b}(\delta)$–defined in $\mathring{U}_2^1(\delta)$ by

$$M(x,\vec{c},\alpha,\vec{\gamma}) \iff \begin{cases} G(z,\vec{c},\alpha,\vec{\gamma}) & \text{if } x=<2,z> \\ A(\vec{y},\vec{c},\vec{\gamma}) & \text{if } x=<1,<y_1,\ldots,y_k>> \\ 0=1 & \text{otherwise} \end{cases}$$

In other words, $\{x\}M$ is equal to $<DEARY_k(\{\vec{y}\}A),\{x\}G>$. It now follows from Lemma 8 that

$$\mathring{U}_2^1(\delta)\vdash \mathit{Witness2}_{\bigwedge\Gamma}^{\vec{c},\vec{\gamma}}(\alpha,\vec{c},\vec{\gamma})\supset \mathit{Witness2}_E^{\vec{c},\vec{\gamma}}(\{x\}M(x,\vec{c},\alpha,\vec{\gamma}),\vec{c},\vec{\gamma}).$$

It remains to show that M is a PSPACE($\alpha,\vec{\gamma}$) predicate. Since G is a PSPACE($\alpha,\vec{\gamma}$) predicate by the induction hypothesis, it suffices to show that A is a PSPACE($\vec{\gamma}$) predicate. But this follows from the fact that A is in $\Sigma_0^{1,b}(\delta)$ and, as we remarked earlier, every relational appearing in A is a PSPACE($\vec{\gamma}$) predicate.

Case (16): ($\Sigma_1^{1,b}(\delta)$*-PIND*). Suppose the last inference of P is

$$\frac{B(\lfloor\tfrac{1}{2}a\rfloor),\Gamma^* \longrightarrow B(a),\Delta^*}{B(0),\Gamma^* \longrightarrow B(t),\Delta^*}$$

where B is a $\Sigma_1^{1,b}(\delta)$–formula and a is the eigenvariable and does not appear in the lower sequent. We shall assume that $B(0)$ is in Γ and $B(t)$ is in Δ. The other cases are easier and are omitted.

Let D be the formula $B(\lfloor\frac{1}{2}a\rfloor)\wedge(\bigwedge\Gamma^*)$, and let $E(\vec{c},a)$ be $B(a)\vee(\bigvee\Delta^*)$, let F be $B(0)\wedge(\bigwedge\Gamma^*)$ and let A be $B(t)\vee(\bigvee\Delta^*)$. The induction hypothesis is that there is a PSPACE$(\alpha,\vec{\gamma})$ predicate G such that G is $\Delta_1^{1,b}(\delta)$-defined by $\mathring{U}_2^1(\delta)$ and such that

$$\mathring{U}_2^1(\delta)\vdash \mathit{Witness2}_D^{\vec{c},a,\vec{\gamma}}(\alpha,\vec{c},a,\vec{\gamma})\supset \mathit{Witness2}_E^{\vec{c},a,\vec{\gamma}}(\{x\}G(x,\vec{c},a,\alpha,\vec{\gamma}),\vec{c},a,\vec{\gamma}).$$

By Lemma 9, there is a term $s(\vec{c},a)$ such that

$$\mathring{U}_2^1(\delta)\vdash(\forall x\le s(\vec{c},a))(\alpha(z)\leftrightarrow\beta(z))\supset$$
$$\supset[\mathit{Witness2}_E^{\vec{c},a,\vec{\gamma}}(\alpha,\vec{c},a,\vec{\gamma})\leftrightarrow \mathit{Witness2}_E^{\vec{c},a,\vec{\gamma}}(\beta,\vec{c},a,\vec{\gamma})].$$

By Lemma 13, there is a $\Delta_1^{1,b}(\delta)$-definable predicate K of $\mathring{U}_2^1(\delta)$ which satisfies

$$K(x,\vec{c},a,\alpha,\vec{\gamma}) \iff \begin{cases} 0=1 & \text{if } x>s(\vec{c},a) \\ \alpha(x) & \text{if } b=0\wedge x\le s(\vec{c},a) \\ K(x,\vec{c},\lfloor\frac{1}{2}a\rfloor,\alpha,\vec{\gamma}) & \\ \quad\text{if } x\le s(\vec{c},a)\wedge \mathit{Witness2}_{\bigvee\Delta^*}(\boldsymbol{\beta}(2,\{x\}K(x,\vec{c},\lfloor\frac{1}{2}a\rfloor,\alpha,\vec{\gamma})),\vec{c},\vec{\gamma}) & \\ G(x,\vec{c},a,<\boldsymbol{\beta}(1,\{x\}K(x,\vec{c},\lfloor\frac{1}{2}a\rfloor,\alpha,\vec{\gamma})),\boldsymbol{\beta}(2,\alpha)>,\vec{\gamma}) & \\ \qquad\qquad\text{otherwise} & \end{cases}$$

Furthermore, by Lemma 13, K is in PSPACE$(\alpha,\vec{\gamma})$. From the definition of K it is readily seen that

$$\mathring{U}_2^1(\delta)\vdash \mathit{Witness2}_F^{\vec{c},\vec{\gamma}}(\alpha,\vec{c},\vec{\gamma})\wedge \mathit{Witness2}_E^{\vec{c},a,\vec{\gamma}}(\{x\}K(x,\vec{c},\lfloor\frac{1}{2}a\rfloor,\alpha,\vec{\gamma}),\vec{c},\lfloor\frac{1}{2}a\rfloor,\vec{\gamma})\supset$$
$$\supset \mathit{Witness2}_E^{\vec{c},a,\vec{\gamma}}(\{x\}K(x,\vec{c},a,\alpha,\vec{\gamma}),\vec{c},a,\vec{\gamma}).$$

Hence it follows by $\Sigma_1^{1,b}(\delta)$-*PIND* that

$$\mathring{U}_2^1(\delta)\vdash \mathit{Witness2}_F^{\vec{c},\vec{\gamma}}(\alpha,\vec{c},\vec{\gamma})\supset \mathit{Witness2}_A^{\vec{c},\vec{\gamma}}(\{x\}K(x,\vec{c},t,\alpha,\vec{\gamma}),\vec{c},\vec{\gamma}).$$

So we define $M(x,\vec{c},\alpha,\vec{\gamma})$ by

$$M(x,\vec{c},\alpha,\vec{\gamma}) \iff K(x,\vec{c},t,\alpha,\vec{\gamma})$$

and M satisfies the conditions of Theorem 12.

Q.E.D. □

10.6. Only EXPTIME is $\Sigma_1^{1,b}$-definable in V_2^1.

Theorem 1 asserted that every EXPTIME function of polynomial growth rate is $\Sigma_1^{1,b}$-definable by V_2^1. The converse is also true. Since the proof of the converse to Theorem 1 is very similar to the arguments in §10.5 concerning $\Sigma_1^{1,b}$-definable functions of U_2^1 and $\mathring{U}_2^1(\delta)$ we shall merely state the results without giving detailed proofs.

Theorem 14: Suppose $A(\vec{c},d)$ is a $\Sigma_1^{1,b}(\delta)$-formula where $\vec{c}$ and d are all the free variables of A. Also suppose $\mathring{V}_2^1(\delta) \vdash (\forall\vec{x})(\exists y)A(\vec{x},y)$. Then there is a $\Delta_1^{1,b}(\delta)$-formula B, a term t and a function f so that

(1) $\mathring{V}_2^1(\delta) \vdash (\forall\vec{x})(\forall y)(B(\vec{x},y) \supset A(\vec{x},y))$

(2) $\mathring{V}_2^1(\delta) \vdash (\forall\vec{x})(\exists y \leq t)B(\vec{x},y)$

(3) $\mathring{V}_2^1(\delta) \vdash (\forall\vec{x})(\forall y)(\forall z)(B(\vec{x},y) \wedge B(\vec{x},z) \supset y=z)$

(4) For all $\vec{n}$, $\mathbf{N} \models B(\vec{n},f(\vec{n}))$

(5) f is an EXPTIME function

Hence, f is an EXPTIME function which is $\Sigma_1^{1,b}(\delta)$-definable in $\mathring{V}_2^1(\delta)$ and for all $\vec{n}$, $A(\vec{n},f(\vec{n}))$ is true.

The converse to Theorem 1 is an immediate corollary of Theorem 14:

Corollary 15: Suppose $A(\vec{c},d)$ is a $\Sigma_1^{1,b}$-formula where $\vec{c}$ and d are all the free variables of A. Also suppose $V_2^1 \vdash (\forall\vec{x})(\exists y)A(\vec{x},y)$. Then there is an EXPTIME function f such that for all $\vec{n}$, $\mathbf{N} \models A(\vec{n},f(\vec{n}))$.

As before, the proof of Theorem 14 is based on a more complicated theorem:

Theorem 16: Suppose $\mathring{V}_2^1(\delta) \vdash \Gamma,\Pi \longrightarrow \Lambda,\Delta$ and each formula in $\Gamma \cup \Delta$ is a $\Sigma_1^{1,b}(\delta)$-formula and each formula in $\Pi \cup \Lambda$ is a $\Pi_1^{1,b}(\delta)$-formula. Let $c_1,\ldots,c_p$ and $\gamma_1,\ldots,\gamma_q$ be the free variables in $\Gamma,\Pi \longrightarrow \Lambda,\Delta$. Let X and Y be the $\Sigma_1^{1,b}$-formulae

$$X = (\bigwedge\Gamma)\wedge\bigwedge\{\neg C : C\in\Lambda\}$$

and

$$Y = (\bigvee\Delta)\vee\bigvee\{\neg C : C\in\Pi\}.$$

Then there is an EXPTIME($\alpha,\vec{\gamma}$) predicate M so that

(1) M is $\Delta_1^{1,b}(\delta)$–defined by $\mathring{V}_2^1(\delta)$ and

(2) $\mathring{V}_2^1(\delta)\vdash \mathit{Witness2}_X^{\vec{c},\vec{\gamma}}(\alpha,\vec{c},\vec{\gamma})\supset \mathit{Witness2}_Y^{\vec{c},\vec{\gamma}}(\{x\}M(x,\vec{c},\alpha,\vec{\gamma}),\vec{c},\vec{\gamma})$.

The proof of Theorem 16 is almost exactly like the proof of Theorem 12. The only substantive difference is in Case (16), where the last inference of P is a $\Sigma_1^{1,b}(\delta)$–*IND* inference. In this case, instead of using Lemma 13 we use Lemma 17:

Lemma 17: Let $A(a,\vec{c},\vec{\gamma})$ and $B(a,b,\vec{c},\alpha,\vec{\gamma})$ be $\Delta_1^{1,b}(\delta)$–formulae of $\mathring{V}_2^1(\delta)$ where α is a unary predicate variable. Let $t(b,\vec{c})$ be a term with only the free variables b and $\vec{c}$ as indicated. Let $K(a,b,\vec{c},\vec{\gamma})$ be defined from A and B as in Theorem 9.14 by:

$$K(a,b,\vec{c},\vec{\gamma}) \iff \begin{cases} A(a,\vec{c},\vec{\gamma}) & \text{if } b=0 \text{ and } a\le t(b,\vec{c}) \\ 0=1 & \text{if } a>t(b,\vec{c}) \\ B(a,b,\vec{c},\{x\}K(x,b\dot{-}1,\vec{c},\vec{\gamma}),\vec{\gamma}) & \text{otherwise} \end{cases}$$

Then $K(a,b,\vec{c},\vec{\gamma})$ is $\Delta_1^{1,b}$–definable by $\mathring{V}_2^1(\delta)$. Furthermore, if A is in EXPTIME($\vec{\gamma}$) and B is in EXPTIME($\alpha,\vec{\gamma}$) then K is in EXPTIME($\vec{\gamma}$).

Proof: The proof of Theorem 9.14 shows that $K(a,b,\vec{c},\vec{\gamma})$ is $\Delta_1^{1,b}(\delta)$–defined by $\mathring{V}_2^1(\delta)$. If A is EXPTIME($\vec{\gamma}$) computable and B is EXPTIME($\alpha,\vec{\gamma}$) computable, then the straightforward algorithm for computing $K(a,b,\vec{c},\vec{\gamma})$ is an EXPTIME($\vec{\gamma}$)–algorithm.

Q.E.D. □

10.7. A Corollary about NEXPTIME∩ co-NEXPTIME.

Definition: NEXPTIME is the set of predicates which are recognized by a non-deterministic exponential time Turing machine. The set co–NEXPTIME is the set of predicates whose complements are in NEXPTIME.

Proposition 18: A predicate $Q(\vec{x})$ is in NEXPTIME iff there is a formula $A \in \Sigma_1^{1,b}$ such that

$$Q(\vec{x}) \iff \mathbf{N} \models A(\vec{x}).$$

Proof: By Corollary 9.17, every $\Sigma_1^{1,b}$-formula $A(\vec{x})$ is equivalent to a formula of the form $(\exists\phi)B(\vec{a},\phi)$ where $B \in \Sigma_0^{1,b}$. By Lemma 9, there is a term $s_B(x)$ so that the value of $B(\vec{x},\phi)$ only depends on the values of $\phi(y)$ for $y \le s_B(\vec{x})$. Thus a $\Sigma_1^{1,b}$-formula $A(\vec{x})$ can be evaluated in nondeterministic exponential time by first guessing the values of $\phi(y)$ for all $y \le s_B(\vec{x})$ and then evaluating $B(\vec{x},\phi)$.

Conversely, it follows from the methods of §10.1 that every NEXPTIME predicate $Q(\vec{x})$ can be expressed by a $\Sigma_1^{1,b}$-formula $A(\vec{x})$. If M is a nondeterministic Turing machine which computes $Q(\vec{x})$ in time $2^{q(|\vec{x}|)}$, let $A(\vec{x})$ be $(\exists\lambda^n)Run_M(\lambda^n, 2^{q(|\vec{x}|)}, \vec{x})$.

Q.E.D. □

Corollary 19:

(a) If $A(\vec{x})$ is any formula and U_2^1 proves $A(\vec{x})$ is equivalent to a $\Sigma_1^{1,b}$– and a $\Pi_1^{1,b}$–formula then $A(\vec{x})$ represents a predicate in PSPACE. In other words, if U_2^1 proves A is in NEXPTIME∩co–NEXPTIME then $A \in$PSPACE.

(b) If $A(\vec{x})$ is any formula and V_2^1 proves $A(\vec{x})$ is equivalent to a $\Sigma_1^{1,b}$– and a $\Pi_1^{1,b}$–formula then $A(\vec{x})$ represents a predicate in EXPTIME. In other words, if V_2^1 proves A is in NEXPTIME∩co–NEXPTIME then $A \in$EXPTIME.

Proof: This is just a restatement of Corollaries 11 and 15. The proof is similar to the proof of Theorem 5.9 and Corollary 5.10. □

Corollary 19 also holds for the theories $\mathring{U}_2^1$ and $\mathring{V}_2^1$.

10.8. Variations, Complications and Open Questions.

Some questions concerning second order Bounded Arithmetic which have not been resolved include:

(1) Is V_2^1 equivalent to U_2^1?

(2) Is $\tilde{U}_2^1$ equivalent to $\mathring{U}_2^1$? Is $\tilde{V}_2^1$ equivalent to $\mathring{V}_2^1$?

(3) Is U_2^1 or V_2^1 a conservative extension of S_2?

(4) Is U_2^1 or V_2^1 a conservative extension of S_2^i?

The author conjectures that the answers to questions (1), (3) and (4) are "no". In particular, if (1) has an affirmative answer, then PSPACE=EXPTIME.

Corollary 20: If $U_2^1 \equiv V_2^1$ then PSPACE=EXPTIME. Also, if $\hat{U}_2^1(\delta) \equiv \hat{V}_2^1(\delta)$ then PSPACE=EXPTIME.

Proof: By Theorems 1, 4, 10 and 14. □

However, there seems to be no reason why U_2^1 could not be a conservative extension of S_2^1. There is no evidence that this would imply P=PSPACE, for instance.

A topic for further research would be to investigate the theories U_2^i and V_2^i for $i>1$. It would be nice to establish what functions can be $\Sigma_i^{1,b}$–defined in these theories. It appears that the $\Sigma_i^{1,b}$–defined functions of V_2^i are precisely the functions at the i-th level of the exponential time hierarchy. That is, V_2^2 can $\Sigma_2^{1,b}$–define precisely the functions which can be computed by an exponential time Turing machine using an oracle for a NEXPTIME-complete predicate, etc. The situation for U_2^i is not quite as clear. First of all, computer scientists do not recognize a polynomial space hierarchy: a well known theorem of Savitch [24] states that PSPACE=NPSPACE. Instead we expect that U_2^i can $\Sigma_i^{1,b}$–defined precisely the function which can be computed by a polynomial space bounded Turing machine using an oracle from the i-th level of the exponential time hierarchy. For example, we expect that U_2^2 can $\Sigma_2^{1,b}$–define precisely the functions which can be computed by a polynomial space bounded Turing machine with an oracle for a NEXPTIME-complete predicate.

A variation of second order Bounded Arithmetic is to restrict all predicate and function variables to have bounded domains. A predicate ϕ has bounded domain iff there is a z such that when $x_i>z$ for some x_i then $\phi(\vec{x})$ does not hold. Likewise, a function ς has bounded domain iff there is a z such that when some $x_i>z$, $\varsigma(i)$=0.

We change the second order language so that the second order predicate variables are α_i^s and ϕ_i^s and the second order function variables are $\varsigma_i^{t,s}$ and $\lambda_i^{t,s}$ where s and t are arbitrary terms. Let $\vec{x}$ be a list of new variables not appearing in s. Then second order Bounded Arithmetic contains the new axioms

$$(\forall\vec{x})(\forall\phi_i^s)((\bigvee x_i>s)\supset\neg\phi_i^s(\vec{x}))$$

$$(\forall\vec{x})(\forall\lambda_i^{t,s})((\bigvee x_i>s)\supset\lambda_i^{t,s}(\vec{x})=0)$$

$$(\forall\vec{x})(\forall\lambda_i^{t,s})(\lambda_i^{t,s}(\vec{x})\leq t).$$

Thus the axioms force all predicate and function variables to range over bounded domain predicates and functions.

We also change the comprehension axioms for bounded domains. The bounded domain comprehension axioms (the $\Sigma_0^{1,b}$–BCA axioms) are

$$(\forall\vec{z})(\forall\vec{\phi}^q)(\forall\vec{\eta}^{t,s})(\exists\lambda^r)(\forall\vec{y}\le r)[\lambda^r(\vec{y})\leftrightarrow A(\vec{y},\vec{z},\vec{\phi}^q,\vec{\eta}^{t,s})]$$

where $A\in\Sigma_0^{1,b}$. The $\Sigma_1^{1,b}$–$BFCA$, the bounded domain function comprehension axioms are defined similarly. We leave it to the reader to formulate the bounded domain comprehension inferences.

Let $U_2^i(BD)$ and $V_2^i(BD)$ be the theories which use bounded domain predicate and function variables, have the $\Sigma_i^{1,b}$–$PIND$ and $\Sigma_i^{1,b}$–IND (respectively) axioms, and have the $\Sigma_0^{1,b}$–comprehension axioms. So $U_2^i(BD)$ and $V_2^i(BD)$ are similar to U_2^i and V_2^i except they are restricted to using only bounded domain second order variables. It turns out that the same functions are $\Sigma_1^{1,b}$–definable in $U_2^1(BD)$ and $V_2^1(BD)$ as in U_2^1 and V_2^1 respectively; namely the PSPACE and EXPTIME functions (respectively). This is true because the proofs of Theorems 1 and 4 only used functions with bounded domain.

The theories $\tilde{U}_2^i(BD)$ and $\tilde{V}_2^i(BD)$ are defined to be $U_2^i(BD)$ and $V_2^i(BD)$, respectively, restricted to contain only second order predicate variables and no second order function variables. Of course, the analogues of Theorem 9.5 and Lemma 9.6 hold, so $U_2^i(BD)$ and $V_2^i(BD)$ are conservative extensions of $\tilde{U}_2^i(BD)$ and $\tilde{V}_2^i(BD)$, respectively.

As a final topic we discuss the predicativity of second order Bounded Arithmetic. Ed Nelson [19] defines a theory to be ***predicative*** if it can be interpreted in R. Robinson's induction-free, open theory of arithmetic Q. Independently, A. Wilkie and E. Nelson have shown that bounded induction is predicative; in particular, the theories S_2^i and S_2 are predicative.

Second order bounded domain Bounded Arithmetic is also predicative. To show this it suffices to interpret $\tilde{U}_2(BD)$ in the first order theory S_2. So let M be a model of S_2; we construct from M a model N for $\tilde{U}_2^{(-1)}(BD)$. N will consist of two parts N_1 and N_2; both N_1 and N_2 are subsets of the universe of M and N_1 is the first order part of N and N_2 is the second order elements of N. If $\alpha\in N_2$ and $\vec{x}\in N_1$ then we interpret $\alpha(\vec{x})$ in S_2^1 as

$$\beta(<\vec{x}>,\alpha)\neq 0$$

where $<\vec{x}>$ is the sequence coding $x_1,\ldots,x_n$ and satisfies

$$UniqSeq(<\vec{x}>)\wedge\beta(0,<\vec{x}>)=n\wedge(\forall i<n)(\beta(i+1,<\vec{x}>)=x_i).$$

By the results of Chapter 2, it is clear that for each $n>0$ the map $\vec{x}\mapsto<\vec{x}>$ is Σ_1^b–defined by S_2^1. Hence the interpretation of $\alpha(x)$ is well-defined.

Next define $I(x)$ to specify an initial segment of M satisfying

$$I(x) \iff (\exists z)(\exists y)(|z|=y \wedge |y|=x).$$

We let I denote the elements m of M satisfying $I(m)$. So if M is closed under exponentiation $I=M$. Otherwise I is the initial segment of M containing all the m such that 2^{2^m} exists in M. Since, $2^{2^{x+1}}=(2^{2^x})^2$, I is *inductive*; that is, if $m \in I$ then $m+1 \in I$. Using techniques due originally to R. Solovay and independently to E. Nelson, we can find another definable initial segment N_1 of M such that $N_1 \subseteq I$ and N_1 is closed under successor, addition, multiplication, and smash (#). We let $N_2=M$.

We claim that $N=<N_1,N_2>$ is a model of $\tilde{U}_2^{(-1)}(BD)$. This is because the $\Sigma_0^{1,b}$–BCA comprehension axioms can be proved using the $\Sigma_0^{1,b}$–IND axioms of S_2. Since this is straightforward, we omit the proof.

The above shows that $\tilde{U}_2^{(-1)}(BD)$ can be interpreted in S_2. It remains to show that $\tilde{U}_2(BD)$ is interpretable in $\tilde{U}_2^{(-1)}(BD)$. The fact that $\tilde{U}_2(BD)$ can be locally interpreted in $\tilde{U}_2^{(-1)}(BD)$ follows again by the techniques of Solovay and Nelson. (A theory H is *locally interpretable* in another theory G iff any subtheory generated by a finite subset of the axioms of H is interpretable in G.) The fact that $\tilde{U}_2(BD)$ can be globally interpreted in $\tilde{U}_2^{(-1)}(BD)$ follows from a technique due to Wilkie, see Pudlak [22].

If M is not closed under exponentiation, the above construction will actually yield a model N of $\tilde{U}_2$. Hook [16] uses the assumption $(\exists y)(\forall z)(|z|<y)$ as a predicative assumption. Hence, if we accept Hook's axiom as predicative, the (unbounded domain) second order theories U_2^i of Bounded Arithmetic are predicative.

As a corollary to the above discussion we deduce that the PSPACE and EXPTIME functions can be predicatively defined.

POSTSCRIPT

Since the original version of this dissertation appeared, a year ago as of this writing, a number of further developments in Bounded Arithmetic have occurred.

A. Wilkie in a handwritten manuscript titled "A model theoretic proof of Buss's characterization of the polynomial time computable functions" has given a model theoretic proof of a variant of the Main Theorem 5.2 for the case $i=1$. His method of proof readily extends to all $i \geq 1$.

J. P. Ressayre in a handwritten manuscript titled "A conservation result for systems of Bounded Arithmetic" has examined a strong form of the Σ_n^b–replacement axioms and investigated its strength relative to the axioms investigated in Chapter 2 above.

In a paper "The polynomial hierarchy and intuitionistic Bounded Arithmetic" in *Structure in Complexity Theory*, Springer-Verlag Lecture Notes in Computer Science #223, I have extended the Main Theorem of Chapter 5 to intuitionistic theories.

Peter Clote and Gaisi Takeuti in a paper titled "Exponential time and Bounded Arithmetic" in the same volume have extended the Theorems 10.1 and 10.14 to functions which are n-fold exponential time computable. They utilized many-sorted theories of Bounded Arithmetic rather than higher order theories to obtain a more elegant formulation.

However, none of the major open problems concerning Bounded Arithmetic have been solved in the past year. It is hoped that further research will be able to resolve some of them.

BIBLIOGRAPHY

[1] **A. V. Aho, J. E. Hopcraft, J. D. Ullman**, *The design and analysis of computer algorithms*, Addison-Wesley, 1974.

[2] **T. Baker, J. Gill, R. Solovay**, "Relativizations of the P=?NP question," SIAM Journal of Computing **4** (1975) 431-442.

[3] **James Bennet**, *On Spectra*, Ph.D. dissertation, Princeton University, 1962.

[4] **Ashok K. Chandra, Dexter C. Kozen, Larry J. Stockmeyer**, "Alternation," Journal of the ACM **28** (1981) 114-133.

[5] **Alan Cobham**, "The intrinsic computational difficulty of functions," in *Logic, Methodology and Philosophy of Science II*, Jerusalem 1964, pp. 24-30. Edited by Y. Bar-Hillel, North-Holland 1965.

[6] **Stephen Cook**, "Feasibly constructive proofs and the propositional calculus," Seventh ACM Symp. on Theory of Computing (1975) 83-97.

[7] **Steven A. Cook, Robert Reckhow**, "On the lengths of proofs in the propositional calculus," Proc. Sixth ACM Symposium on Theory of Computing, 1974 pp. 135-148.

[8] **Martin Dowd**, personal communication.

[9] **S. Feferman**, "Arithmetization of metamathematics in a general setting," Fundamenta Mathematicae **49** (1960) 35-92.

[10] **Harvey Friedman**, "On the consistency, completeness and correctness problems," unpublished manuscript, June 1979.

[11] **M. Furst, J.B. Saxe, M. Sipser**, "Parity, circuits, and the polynomial-time hierarchy," Proc. 22nd Annual Symp. on the Foundations of Computer Science (1981) 260-270.

[12] **Micheal R. Garey, David S. Johnson**, *Computers and Intractability, A guide to the theory of NP-completeness*, W. H. Freeman, 1979.

[13] **Gerhard Gentzen**, "Untersuchungen über das logische Schliessen," Mathematische Zeitschrift **39** (1935) 176-210, 405-431. English translation in [27].

[14] **C. A. Goad**, "Proofs as descriptions of computations," in *Fifth Conference of Automated Deduction,* Les Arcs, France 1980, pp. 39-52. Springer-Verlag Lecture Notes in Computer Science 87, edited by W. Bibel and R. Kowalski.

[15] **Andrzej Grzegorczyk**, *Some classes of recursive functions*, Rozprawy Matematyczne **4** (1953).

[16] **Jay Hook,** *A many-sorted approach to predicative mathematics*, Ph.D. dissertation, Princeton University, 1983.

[17] **Clarence F. Kent, Bernard R. Hodgson**, "An arithmetical characterization of NP," Theoretical Computer Science **21** (1982) 255-267.

[18] **G. Kreisel**, "Some uses of proof theory for finding computer programmes," *Colloque International de Logique*, Clermont-Ferrand, 1975, pp. 123-134. Colloques Internationaux de Centre National de la Recherche Scientifique no. 249 (1977).

[19] **Edward Nelson**, *Predicative Arithmetic*, manuscript (to appear).

[20] **Rohit J. Parikh**, "Existence and feasibility in arithmetic," Journal of Symbolic Logic **36** (1971) 494-508.

[21] **Jeff Paris, L.A.S. Kirby**, "Σ_n collection schemes in arithmetic," in *Logic Colloquium '77*, North-Holland, 1978, pp. 199-210.

[22] **Pavel Pudlak**, "Some prime elements in the lattice of interpretability types," Transactions of the A.M.S. **280** (1983) 255-275.

[23] **Pavel Pudlak**, "On the length of proofs of finitistic consistency statements in first order theories," *Logic Colloquium '84, Proc. of an ASL Conference in Manchester, England*, North-Holland (to appear).

[24] **W. J. Savitch**, "Relationship between nondeterministic and deterministic tape complexities," Journal of Computer and System Sciences **4** (1970) 177-192.

[25] **Raymond M. Smullyan**, *Theory of Formal Systems*, Annals of Mathematics Studies, no. 47, Princeton University Press, 1961.

[26] **Larry J. Stockmeyer**, "The polynomial-time hierarchy," Theoretical Computer Science **3** (1976) 1-22.

[27] **M. E. Szabo**, *The collected papers of Gerhard Gentzen*, North-Holland 1969.

[28] **Gaisi Takeuti**, *Proof Theory*, North-Holland 1975.

[29] **L. G. Valiant**, "The complexity of computing the permanent," Theoretical Computer Science **8** (1979) 189-201.

[30] **Alex Wilkie**, a talk at Logic Colloquium '84, the ASL European Summer Meeting, Manchester, England, July 1984.

[31] **Alex Wilkie, Jeff Paris**, "On the scheme of induction for bounded arithmetic formulas," *Logic Colloquium '84, Proc. of an ASL Conference in Manchester, England*, North-Holland (to appear).

[32] **George Wilmers**, "Bounded existential induction," Journal of Symbolic Logic **50** (1985) 72-90.

[33] **Celia Wrathall**, "Complete sets and the polynomial time hierarchy," Theoretical Computer Science **3** (1976) 23-33.

SYMBOL INDEX

SUBJECT INDEX

Discard this page