Bounded Arithmetic

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

© Copyright 1985, 1986
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>0. Introduction</td>
<td>1</td>
</tr>
<tr>
<td>1. The Polynomial Hierarchy</td>
<td>6</td>
</tr>
<tr>
<td>1.1. Limited Iteration</td>
<td>6</td>
</tr>
<tr>
<td>1.2. Polynomial-time Computations</td>
<td>10</td>
</tr>
<tr>
<td>1.3. Bounded Quantifiers</td>
<td>13</td>
</tr>
<tr>
<td>1.4. The Polynomial Hierarchy</td>
<td>14</td>
</tr>
<tr>
<td>1.5. Eliminating PTC</td>
<td>16</td>
</tr>
<tr>
<td>1.6. Bounded Arithmetic Formulae</td>
<td>19</td>
</tr>
<tr>
<td>1.7. Relativization of the Polynomial Hierarchy</td>
<td>21</td>
</tr>
<tr>
<td>1.8. Appendix</td>
<td>25</td>
</tr>
<tr>
<td>2. Foundations of Bounded Arithmetic</td>
<td>28</td>
</tr>
<tr>
<td>2.1. The Language of Bounded Arithmetic</td>
<td>28</td>
</tr>
<tr>
<td>2.2. Axiomatizations of Bounded Arithmetic</td>
<td>30</td>
</tr>
<tr>
<td>2.3. Introducing Function and Predicate Symbols</td>
<td>37</td>
</tr>
<tr>
<td>2.4. Bootstrapping $S^1_2$ - Phase 1</td>
<td>37</td>
</tr>
<tr>
<td>2.5. Bootstrapping $S^1_2$ - Phase 2</td>
<td>45</td>
</tr>
<tr>
<td>2.6. Bootstrapping $T^1_2$</td>
<td>50</td>
</tr>
<tr>
<td>2.7. Replacement Axioms</td>
<td>53</td>
</tr>
<tr>
<td>2.8. Minimization Axioms</td>
<td>55</td>
</tr>
<tr>
<td>2.9. Summary of Axiomatizations of Bounded Arithmetic</td>
<td>60</td>
</tr>
<tr>
<td>3. Definability of Polynomial Hierarchy Functions</td>
<td>62</td>
</tr>
<tr>
<td>4. First-Order Natural Deduction Systems</td>
<td>66</td>
</tr>
<tr>
<td>4.1. Syntax and Rules of Natural Deduction</td>
<td>66</td>
</tr>
<tr>
<td>4.2. Bounded Arithmetic</td>
<td>71</td>
</tr>
<tr>
<td>4.3. Cut Elimination</td>
<td>73</td>
</tr>
<tr>
<td>4.4. Further Normal Forms for Proofs</td>
<td>75</td>
</tr>
<tr>
<td>4.5. Restricting by Parameter Variables</td>
<td>77</td>
</tr>
<tr>
<td>4.6. Polynomial Size, Induction Free Proofs</td>
<td>82</td>
</tr>
<tr>
<td>4.7. Parikh's Theorem</td>
<td>83</td>
</tr>
<tr>
<td>5. Computational Complexity of Definable Functions</td>
<td>86</td>
</tr>
<tr>
<td>5.1. Witnessing a Bounded Formula</td>
<td>86</td>
</tr>
<tr>
<td>5.2. The Main Proof</td>
<td>89</td>
</tr>
<tr>
<td>5.3. The Main Theorem for First Order Bounded Arithmetic</td>
<td>90</td>
</tr>
<tr>
<td>5.4. Relativization</td>
<td>100</td>
</tr>
<tr>
<td>6. Cook's Equational Theory PV</td>
<td>104</td>
</tr>
<tr>
<td>6.1. Preliminaries for PV and PV1</td>
<td>104</td>
</tr>
<tr>
<td>6.2. $S^1_2$ and the Language of PV</td>
<td>107</td>
</tr>
</tbody>
</table>
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 LaForty 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 Sloan 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 $A_f$. Now we can show that $f$ is computable (relative to a given model of computation) if and only the language $A_f$ 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 $A_f$.

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 $A_f$ 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 $x$. 

1
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:

\[
\begin{align*}
0 & \quad \text{zero constant symbol} \\
S & \quad \text{successor} \\
+ & \quad \text{addition} \\
\cdot & \quad \text{multiplication} \\
\lfloor x \rfloor & \quad \text{"shift right" function, i.e., divide by two and round down} \\
\lfloor x \rfloor & \quad \text{the length of the binary representation of } x \\
\lfloor x \rfloor y & \quad \text{the "smash" function} \\
\leq & \quad \text{less than or equal to}
\end{align*}
\]

(The notations \([a]\) and \([a]\) 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 \((\forall x \leq t)A\) or \((\exists x \leq t)A\) where \(t\) is any term not involving \(x\). The meanings of \((\forall x \leq t)A\) and \((\exists x \leq t)A\) are \((\forall x)x \leq t \implies A\) and \((\exists x)x \leq t \implies A\), 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 formulæ.

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 formulæ in a hierarchy \(\Sigma_0^b, \Sigma_1^b, \Pi_1^b, \Sigma_2^b, \Pi_2^b, \ldots\) by counting alternations of bounded quantifiers, ignoring the sharply bounded quantifiers. This is analogous to the
Introduction

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 $\Sigma^b_1$-PIND. The $\Sigma^b_1$-PIND axioms are the formulae

$$\forall \Sigma_1^b (A(x) \rightarrow \forall \Sigma_1^b (\exists x A(x))$$

where $A$ is a $\Sigma_1^b$-formula. We define in Chapter 2 a hierarchy of theories $S_2^b$, $S_3^b$, $S_4^b$, ... so that $S_2^b$ is a theory of Bounded Arithmetic axiomatized by a few simple open axioms and by $\Sigma^b_1$-PIND.

If $R$ is a theory of Bounded Arithmetic we say that the function $f \in \Sigma^b_1$-definable in $R$ iff there is a $\Sigma^b_1$-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) $R \vdash (\forall x)(\forall y)(\exists z)(z=y)\rightarrow A(x,y)$.

We shall be mostly interested in functions which are $\Sigma^b_1$-definable in $S_2^b$.

The Meyer-Stockmeyer polynomial hierarchy is a hierarchy of predicates on the non-negative integers which can be computed in polynomial time by a generalised 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 $\Sigma^P_1$, 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 $P$, $\Sigma^P_1$, $\Pi^P_1$, $\Sigma^P_2$, $\Pi^P_2$, ...

We can extend the polynomial hierarchy to a hierarchy of functions by defining $\Delta^P_1 = PTC(\Sigma^p_1)$, the Polynomial-Time Closure of $\Sigma^p_1$, 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 $\Sigma^p_1$.

It is well known (and we prove it again in Chapter 1) that the predicates in $\Sigma^p_1$ are precisely the predicates which can be expressed by a $\Sigma^b_1$-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 $\Sigma^b_1$-definable in $S^b_2$ is a $\Omega^b_1$-function, and conversely that every $\Omega^b_1$-function is $\Sigma^b_1$-definable in $S^b_2$. (See Theorem 5.6 for the strongest version of this theorem.) This provides a characterization of the functions which are $\Sigma^b_1$-definable in $S^b_2$ in terms of computational complexity.
The hardest part of this theorem is showing that every $\Sigma^1_4$-definable function is in $\Sigma^1_4$. An extremely brief outline of the proof is as follows: Let $A$ be a $\Sigma^1_4$-formula and suppose $S^2_4$ proves $(\forall y)(\exists z)A(z,y)$. By Gentzen’s cut elimination theorem there is a free cut free proof of $(\forall y)(\exists z)A(z,y)$. By examining the allowable inferences of natural deduction we discover that this free cut free proof contains an explicit $\Sigma^1_4$-algorithm for computing $y$ from $x$. This method of proof is reminiscent of Kreisel [18] and Conad [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 $S^2_4$ can $\Sigma^1_4$-define precisely the polynomial time functions and $S^2_5$ can $\Sigma^1_4$-define precisely the functions in PTO($NP$).

The import of this theorem is twofold. On one hand, it provides a characterization of the $\Sigma^1_4$-functions in terms of their definability by the formal theory $S^2_4$ of arithmetic. On the other hand, it states that the proof-theoretic strength of the formal theory $S^2_4$ is closely linked to the computational complexity of $\Sigma^1_4$-functions.

Another way to state the main theorem is as follows: if $A \in \Sigma^1_4$ and $B \in \Pi^1_4$ and if $S^2_4 \vdash A \iff B$, then the predicate defined by $A$ and $B$ is in PTO($\Sigma^1_4$). In particular, any predicate which $S^2_4$ cannot prove is equivalent to both a $\Sigma^1_4$ and a $\Pi^1_4$-formula in $\Pi^1_4$; in other words, since $\Sigma^1_4$- and $\Pi^1_4$-formulas represent $NP$ and co-$NP$ predicates, the class of predicates which $S^2_4$ 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 $U^1_2$ and $V^1_2$ of second-order Bounded Arithmetic which have the property that the functions $\Sigma^1_4$-definable in $U^1_2$ (respectively, $V^1_2$) 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 EXP TIME functions in terms of definability in second-order Bounded Arithmetic.

Chapter 7 discusses improved versions of Godel 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 $FCFCon(S^2_1)$ which asserts that there is no free cut free $S^2_1$-proof of a contradiction. Also, there is a formula $BCDCon(S^2_1)$ which asserts that there is no $S^2_1$-proof of $P$ of a contradiction such that every formula in $P$ is bounded. We show that, for $i \geq 1$, $S^2_i$ can prove neither $FCFCon(S^1_i)$ nor $BCDCon(S^2_1)$.

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 8.8 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 [29] 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 \( O((p(n))^3) \), (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 as closure under composition and limited iteration, we can construct all polynomial time computable functions.

We adopt the convention that all functions have domain \( \mathbb{N}^k \) and codomain \( \mathbb{N} \) for the rest of this dissertation where \( \mathbb{N} \) 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 \( \mathbb{N}^k \) to \( \mathbb{N} \):

1. \( 0 \), (the constant zero function)
2. \( x \mapsto Sx \), (the successor function)
3. \( x \mapsto [\frac{1}{2}x] \), (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 \text{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 \( \text{True} \) and if \( x = 0 \) then \( x \) represents \( \text{False} \).

   \begin{align*}
   \neg x &= x \leq 0 = \text{Choice}(x,0,1) \\
   \land (x \land y) &= \text{Choice}(x,y,0) \\
   \lor (x \lor y) &= \text{Choice}(x,1,y) \\
   \oplus (x \oplus y) &= (\neg x \land y) \lor (x \land \neg y)
   \end{align*}

   It is important to note that for the time being \( \neg, \land, \lor \) and \( \oplus \) are numerical operations. Later we will use \( \neg, \land, \lor \) extensively as logical operators.

2. Equality and Inequality:

   \begin{align*}
   (x=y) &= (x \leq y) \land (y \leq x) \\
   (x \neq y) &= (x \leq y) \land \neg (y \leq x)
   \end{align*}

3. Arithmetic modulo 2:

   \begin{align*}
   (x \equiv 2) &= \neg (x \equiv 2 \cdot [\frac{1}{2}x])
   \end{align*}

   \( (x \equiv 2) \) 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 \( \lambda \)
Definition: \( B^+ \) is the set of functions which contain all the functions in \( B \) plus the following functions:

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

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

2. \( \text{Truncate}(\langle a_1, a_2, \ldots, a_n \rangle) = \langle a_2, \ldots, a_n \rangle \)

3. \( \alpha^* < a_1, \ldots, a_n > = < a_0, a_1, \ldots, a_n > \)

Again, the values of the functions \( \text{Truncate} \) and \( \alpha \) 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 \( x \) the length of the binary representation of \( x \). Note that \( |0| = 0 \).

If \( \mathcal{Z} \) is a vector of numbers \( x_1, \ldots, x_n \) then \( |\mathcal{Z}| \) denotes the vector \( |x_1|, \ldots, |x_n| \).

Definition: \( p \) is a suitable polynomial if \( p \) has non-negative integer coefficients.

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

Let \( r : N^{k+1} \rightarrow N \) be defined as:

\[
\begin{align*}
\tau(x_1, \ldots, x_k, 0) &= g(x_1, \ldots, x_k) \\
\tau(x_1, \ldots, x_k, n+1) &= h(x_1, \ldots, x_k, n, \tau(x_1, \ldots, x_k, n)).
\end{align*}
\]

Then we must have

\[
(\forall n \leq |\mathcal{Z}|)(|\tau(\mathcal{Z}, n)| \leq q(|\mathcal{Z}|))
\]

and \( f(\mathcal{Z}) \) is defined by
§1.1 Limited Iteration

\[ f(x) = r(x, p(x)) \]

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

**Definition:** A function \( f: \mathbb{N}^k \to \mathbb{N} \) has polynomial growth rate if there is a suitable polynomial \( p \) such that for all \( x \), we have \( |f(x)| \leq p(|x|) \). Let \( C \) be a set of functions of polynomial growth rate. The Polynomial-time closure of \( C \), denoted \( PTC(C) \), 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. \( \square \)

As an illustration of how limited recursion is used, we show that addition is in \( PTC(\emptyset) \).

We first define \( f_\delta(x, y) \) by limited recursion from \( g_1 \) and \( h_1 \) with bounds \( p_1 \) and \( q_1 \), where

\[
\begin{align*}
g_1(x, y) &= 1 + 0 \uparrow x + y \uparrow 0 = <1, 0, x, y> \\
h_1(x, y, m, w) &= SM(\beta(x, y, w), \beta(y, y, w), \beta(x, x, w), \beta(y, y, w), \beta(w, w, 2)) \uparrow \\
&\quad \uparrow \mathbf{CARRY}(\beta(x, y, w), \beta(y, y, w), \beta(x, x, w), \beta(y, y, w), \beta(w, w, 2)) \uparrow \\
&\quad \uparrow \beta(x, x, w) \uparrow \beta(y, y, w) \uparrow 0 \\
p_1(n, m) &= n + m \\
q_1(n, m) &= 2n + 2m + 14
\end{align*}
\]

and where

\[
SM(x, a, b, c) = \text{Choice}(a \oplus b \oplus c, S(2 x), 2 x) \\
\text{CARRY}(a, b, c) = (a \land b) \lor (a \land c) \lor (b \land c).
\]

Note that in the definition of \( g_1 \), the formula \( 1 + 0 \uparrow x + y \uparrow 0 \) means \( 1 + ((0 \uparrow x) + y) \) which is \( <1, 0, x, y> \). Similar considerations apply to the definition of \( h_1 \) and for the rest of Chapter 1 we follow the convention that \( \uparrow \) associates from right to left.

Intuitively, \( f_\delta(x, y) = <\text{FlippedSum}(x, y), 0, 0, 0> \), where \( \text{FlippedSum}(x, y) \) is a number whose binary expansion contains the binary expansion of \( x+y \) in reverse order immediately following the high order bit. For example, \( f_\delta(4, 8) = <1\{0011000\}, 0, 0, 0> \). 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 \( f_\delta \in PTC(\emptyset) \).

Secondly, we define \( f_\delta(x) \) by limited iteration from \( g_2 \) and \( h_2 \) with bounds \( p_2 \) and \( q_2 \), where...
The Polynomial Hierarchy

\[ g(x) = 0 \text{ if } 0 < x \]

\[ h_d(x, m, w) = \text{Choice}(\beta(2, w) \times S(2, \beta(1, w)), 2 \beta(1, w)) \times [\beta(2, w)] \times 1 \]

\[ p_d(n) = n \]

\[ q_d(n) = 2^n + 6. \]

We now define Flip(x) using composition by

\[ \text{Flip}(x) = \beta(1, f_d(x)) \]

and finally we can define addition as

\[ x + y = [\text{Flip}(\beta(1, f_d(x), y))]. \]

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 \( f \in PTC(C) \) if and only if 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 \( h \). 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.

\( \implies \) 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 \( B_{\log} C \) 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

$$p_f([x]) + p([x]) + p_h([x], s([x]), q([x])).$$

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.

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 $q(x)$ 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 $q$ in $PTC(C)$. Let the states of $M$ be $q_0, \ldots, q_{N+1}$ where $q_0$ is the initial state and $q_{N+1}$ 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 $b_j$):
   $$b_{l_1} \cdots b_{l_i} b_0 b_1 \cdots b_{l_k}.$$
2. The contents of the oracle tape (current head position is at $b_{s_i}$):
   $$b_{s_m} \cdots b_{s_i} b_0 b_1 \cdots b_{s_n}.$$
3. The current state $q_n$.

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 $q_{M+i}$, the oracle for $h_i$ is invoked with input value coded by the string $b_{s_m} \cdots b_{s_i}$; the value output by the oracle is coded as a binary string and written on the oracle tape as the string $b_{s_m} \cdots b_{s_i}$.

After invoking an oracle the next state $M$ enters is $q_{N+1}$.

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

$$< s, l_1, \ldots, l_i, s_0, \ldots, s_n >.$$

We define $f$ by the following procedure: we first define functions $Init$, $Next$, and $Decode$, then define $f_2$ by limited iteration from $Init$ and $Next$, and finally define $f(x) = Decode(M(f_2(x)))$. 
**The Polynomial Hierarchy**

*Init* is the function computing the initial state of \( M \) with input \( x \). We first define \( f_1(x) \) by limited iteration from \( g_2 \) and \( h_1 \) with bound \( p_1 \) and \( q_1 \), where

\[
g_1(x) = <0, x> \\
h_1(x, m, w) = [S(\beta(2, w) \circ \beta(1, w)) \uplus \beta(2, w)]^{\uplus 0} \\
p_1(n) = n \\
q_1(n) = 2n + 4.
\]

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

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

\[
g_2(x) = x \circ 0 \circ 0 = <x, 0> \\
h_2(u) = Truncate(\beta(1, w)) \circ Choice(\beta(1, \beta(1, w)) \uplus 1, 2, \beta(2, w), S(2 \cdot \beta(2, w))) \uplus 0 \\
p_2(n) = n \\
q_2(n) = 2n + 4.
\]

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

\[
T_0(x) = Choice(\beta(3, x), \beta(1, \beta(3, x)), 0) \\
S_0(x) = Choice(\beta(5, x), \beta(1, \beta(5, x)), 0) \\
U(x) = \beta(1, x).
\]

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

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

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 \( q_3(x) = s([N + k \cdot J + 2]) \cdot (q(n) + 1) \). So \( q_3 \) bounds the length of the Gödel numbers of ID’s of \( M \). Now define
Polynomial-time Computations

\[ f(x) = \text{Decode}(\varepsilon(3, f_{\varepsilon}(x))) \]

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 \textit{predicate} is a function with range \( \{0,1\} \) where \( 0 \) denotes \textit{false} and \( 1 \) denotes \textit{true}.

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

\textbf{Definition:} Let \( Q \) and \( R \) be functions. Then \( (\forall y \leq Q(x))R(x,y) \) is the predicate (i.e., function of \( x \)) which has value 1 iff for all \( y \leq Q(x) \) the value of \( R(x,y) \) is nonzero. Similarly, \( (\exists y \leq Q(x))R(x,y) \) is the predicate which has value 1 iff for some \( y \leq Q(x) \) the value of \( R(x,y) \) 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 \( \Delta^b_0 \) 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. \textit{Polynomially Bounded Quantification} allows bounds of the form \( 2^p(n) \), where \( p \) is a polynomial; whereas \textit{Logarithmically Bounded Quantification} allows only bounds of the form \( p(|t|) \).

\textbf{Definition:} Let \( C \) be a set of functions closed under composition. Then \( PBQ(C) \) is the set of predicates \( Q \) such that

(1) \( Q.N \rightarrow N \) for some \( \varepsilon \in N; \)

(2) There is an \( RC\text{PRED}(C) \) and a suitable polynomial \( p \) such that for all \( x, \)

\[ Q(x) = (\exists y \leq 2^{p(|x|)})R(x,y) \]

\( PBV \) is defined similarly with a universal quantifier replacing the existential quantifier in (2). Note that \( PBV(C) \) and \( PBQ(C) \) 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. \( Q: \mathbb{N}^i \to \mathbb{N} \) for some \( i \in \mathbb{N} \);
2. There is an \( R \in \text{PRED}(C) \) and a suitable polynomial \( p \) such that for all \( \bar{x} \),

\[
Q(\bar{x}) = (\exists y \leq p(||\bar{x}||)) R(\bar{x}, y).
\]

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

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 Beanes [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.

**Definitions:** (by induction on \( k \))

1. \( L_k^f \) is the smallest set of functions containing \( B \) and closed under composition, \( LE \) and \( LB\forall \).
2. \( \Delta_k^f = \Sigma_k^f = \Pi_k^f = \text{PRED}(L_k^f) \).
3. \( \Delta_{k+1}^f = \text{PTC}(\Sigma_k^f) \).
4. \( \Sigma_{k+1}^f = \text{PRED}(\Delta_k^f) \).
5. \( \Pi_{k+1}^f = \text{PBO}(\Delta_k^f) \).
6. \( PH = \bigcup_k \Sigma_k^f \).

The sets of predicates \( \Delta_k^f, \Sigma_k^f \) and \( \Pi_k^f \) are well known to computer scientists and are called \( P, NP \) and \( \text{co-NP} \) respectively. Figure 1 shows a diagram of the hierarchy of predicates \( \Delta_k^f, \Sigma_k^f \) and \( \Pi_k^f \).
Proposition 4: $\Pi_{i+1}^P = P TC(\Pi_i^P)$ 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 $\Sigma_i^P = \Sigma_{i+1}^P$. 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 $\Delta_i^P = \Sigma_i^P \cap \Pi_i^P$? 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 effort has failed to resolve these questions.
One question we can answer is whether $\Delta^P_f = \Delta_f^P$.

**Proposition 5:** $\Delta_f^P \neq \Delta_f^P$.

**Proof:** Let Parity: $\mathbb{N} \rightarrow \mathbb{N}$ be the function defined as

$$\text{Numones}(x) = \# \text{ of ones in the binary representation of } x$$

$$\text{Parity} (x) = \text{Numones}(x) \mod 2.$$

Clearly, Parity $\in \Delta_f^P = \overline{P}$. So it suffices to show that Parity $\notin \Delta_f^P$.

It is easy to show that if $f \in \Delta_f^P$ 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 Fürst, 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 $P \overline{B}$ (polynomially bounded quantification). It turns out that the use of PTC is unnecessary and that the classes $\Sigma_f^P$ and $\Pi_f^P$ can be defined without using PTC and hence without using either Turing machines or limited iteration.

**Lemma 5:**

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

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

**Proof:**

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

$$Q(x) = \left( \forall z \leq p(|x|) R(x, z) \right).$$

We can define $Q(x)$ by limited iteration from $g$ and $h$ with bounds $p$ and $q$, where...
\[ g(\mathcal{F}) = R(\mathcal{F}, w) \]
\[ h(\mathcal{F}, \mathcal{M}, y) = g \circ R(\mathcal{F}, \mathcal{M}) \]
\[ q(n) = 1. \]

Since \( g \) and \( h \) are in \( \Delta_1^2 \), so is \( q \). This shows \( \Delta_1^2 \) is closed under \( LB^2 \) and a similar argument shows \( \Pi_1^2 \) is closed under \( LB^2 \).

(b) Since \( \Sigma_1^2 = \Pi_1^2 = \Delta_1^2 \), (b) is just a special case of (a) when \( k = 0 \). So suppose \( k \geq 1 \). The closure of \( \Pi_1^2 \) and \( \Sigma_1^2 \) under conjunction and disjunction follows easily from (a). To show that \( \Sigma_1^2 \) is closed under \( LB^2 \) it suffices to show that if \( R \in \Delta_1^2 \), and if \( p \) and \( q \) are suitable polynomials, then

\[ S(\mathcal{F}) = (\forall \mathcal{I} \in \mathcal{F})(\exists \mathcal{I} \in (\mathcal{F}, \mathcal{M}, y, z) \]

is in \( \Sigma_1^2 \). But \( S(\mathcal{F}) \) is equivalent to

\[ (\exists \mathcal{I} \in (\mathcal{F}, \mathcal{M}, y, z) (\forall \mathcal{I} \in (\mathcal{F}, \mathcal{M}, y, z)) R(\mathcal{I}, \mathcal{M}, y, z) \]

where \( r(\mathcal{F}) = 2 \cdot (q(\mathcal{I}(\mathcal{M}), \mathcal{I}(y), +2) - q(\mathcal{I}(\mathcal{M}), \mathcal{I}(y), +1) \). Thus \( S(\mathcal{F}) \) is in \( \Sigma_1^2 \). A similar argument shows \( \Pi_1^2 \) is closed under \( LB^2 \).

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 \( k \geq 1 \), \( \Sigma_1^{k+1} = PB^k(\Pi_1^k) \) and \( \Pi_1^{k+1} = PB^k(\Sigma_1^k) \).

(b) Let \( B^* \) be the smallest set containing \( B^* \) which is closed under \( LB^i \), \( LB^2 \), and composition. Then \( \Sigma_1^k = PB^k(\Sigma_1^k) \) and \( \Pi_1^k = PB^k(\Pi_1^k) \).

**Proof:** In order to prove (a) and (b) simultaneously, we define \( D_{k+1} \) to be \( \Pi_1^{k+1} \) and \( E_{k+1} \) to be \( \Sigma_1^{k+1} \), and \( D_0 = E_0 \) to be \( B^* \). The theorem asserts that \( \Sigma_1^{k+1} = PB^k(D_k) \) and \( \Pi_1^{k+1} = PB^k(E_k) \) for all \( k \geq 0 \). It suffices to show that \( \Sigma_1^{k+1} = PB^k(D_k) \) since \( \Pi_1^{k+1} = PB^k(E_k) \) is an immediate consequence of this.

Let \( k \) be a fixed nonnegative integer. Directly from the definitions we have \( \Sigma_1^{k+1} \supseteq PB^k(D_k) \). We need to show the reversal 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 \( i \), \( PB^k(D_k) \supseteq PB^k(C_i) \). Since \( \bigcup C_i = \Omega^{1+1} \), this suffices to prove the theorem. We will show by induction on \( k \) that for any \( K \subseteq C_i \),
\[ S(\bar{x}) = (\exists z \leq 2^{2^{\|\bar{x}\|}})(\forall y \leq q(\|\bar{x}\|))Q(\bar{x}, y, z) \]

is in \(PB^2(D_4)\). (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 \(C_0 = D_4\) and by Lemma 6(b). \(D_4\) is closed under \(PB^1\). So assume \(i > 0\). Without loss of generality we may assume \(Q\) has the form

\[ Q(\bar{x}) = G(\bar{x}, F_1(\bar{x}), \ldots, F_k(\bar{x})). \]

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, \(Q(\bar{x}) = G(F_1(F_2(\bar{x})))\) is equivalent to the formula \((\exists \bar{v} \leq 2^{2^{\|\bar{x}\|}})(\forall \bar{w} \leq 2^{2^{\|\bar{x}\|}})F(\bar{w})G(F(\bar{v}))))\), where \(q\) is a suitable polynomial which bounds the function \(F_k\). 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 \(\beta\) function is always in \(D_k\) and hence it is permissible for \(G\) to involve the pairing function.)

Let each \(F_i\) 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 \(ValidComp(w, \bar{x})\) to be True iff

1. \(w\) is a sequence \(<w_1, \ldots, w_n>\) and
2. Each \(w_i\) codes a sequence \(<w_{i,0}, \ldots, w_{i,n_i}>\) which codes the computation of \(F_i(\bar{x})\).

A precise definition is:

\[ ValidComp(w, \bar{x}) = (\beta(0, w) = m) \land \bigwedge_{j=1}^{k} (w_{j,1} = G_j(\bar{x})) \land \bigwedge_{j=1}^{k} (\beta(0, w_j) = p_j(\|\bar{x}\|) + 1) \land \bigwedge_{j=1}^{k} (\forall v \leq \|w_j\|)(\forall u \leq \|\beta(0, w_j)\|)(\forall u' \leq \|w_{j+2}\|)(H_j(\bar{x}, v, w_{j,1})) \]

where we used the abbreviations \(w_j\) for \(\beta(j, w)\) and \(w_{j,1}\) for \(\beta(m, \beta(j, w))\).

Now we can easily find a suitable polynomial \(\tau\) large enough so that \(Q(\bar{x})\) is equivalent to

\[ (\exists \bar{w} \leq 2^{2^{\|\bar{x}\|}})(ValidComp(w, \bar{x}) \land G(\bar{x}, w_1, \ldots, w_n)). \]

The only quantifiers in \(ValidComp\) are logarithmically bounded quantifiers, so we may rewrite this last equation as
§1.5

\[(\exists w \leq 2^{2^{[w]}})(\forall v \leq [w])R(z, x, v, w)\]

where \(R \in C_{L1}\). So \(S(F)\) is equivalent to

\[(\exists x \leq 2^{2^{[x]}})(\forall y \leq [y])R^*(x, x, y^*, y^*)\]

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 \(\beta\) function as a pairing function to contract adjacent like quantifiers (once the \(\beta\) function is in \(D_k\)). Hence \(S(F)\) is equivalent to

\[(\exists x^* \leq 2^{2^{[x]}}, y^* \leq [y])R^*(x^*, y^*)\]

where \(s\) and \(t\) are suitable polynomials and \(R^* \in C_{L1}\). By the induction hypothesis, \(S(F)\) is in \(PB(F(D_k))\), which completes the induction step and the proof.

Q.E.D. □

The point of Theorem 7 is that we now can characterize the classes \(\Sigma^E_k\) and \(\Pi^E_k\) of the polynomial hierarchy in a purely syntactic way. We start with the initial set \(B^E\) of functions and take its closure under composition and logarithmically bounded quantification to obtain \(\mathcal{B}^E\).

We apply polynomially bounded quantification repeatedly to obtain \(\Sigma^E_k\) and \(\Pi^E_k\). (A somewhat stronger result is obtained by Kintzd-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 \(\land, \lor, \rightarrow, \exists, \forall, \supset\), and \(-\) and the non-logical symbols 0, 5, +, ·, \#, \([x]\), \([\frac{1}{2}x]\), and \(\leq\).

The non-logical symbols have the following meanings:

- 0: zero constant symbol
- \(S\): successor
- +: addition
- -: multiplication
- \([\frac{1}{2}x]\): "shift right" function
- \([x]\): \(\lceil \log_2(x+1) \rceil\), the length of the binary representation of \(x\)
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 $[\exists x]$ functions can be used to form a term equal to $2^{[p]}$. 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 if 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. $\Pi^b_0 = \Sigma^b_0 = \Delta^b_0$ is the set of formulae all of whose quantifiers are sharply bounded.

2. $\Sigma^b_{k+1}$ is defined inductively by:
   - (a) $\Sigma^b_{k+1} \supseteq \Pi^b_k$
   - (b) If $A$ is in $\Sigma^b_{k+1}$ then so are $(\exists x \leq t)A$ and $(\forall x \leq [t])A$.
   - (c) If $A, B \in \Sigma^b_{k+1}$, then $A \land B$ and $A \lor B$ are in $\Sigma^b_{k+1}$.
   - (d) If $A \in \Sigma^b_{k+1}$ and $B \in \Pi^b_{k+1}$ then $\neg A$ and $B \supset A$ are in $\Sigma^b_{k+1}$.

3. $\Pi^b_{k+1}$ is defined inductively by:
   - (a) $\Pi^b_{k+1} \supseteq \Sigma^b_k$
   - (b) If $A$ is in $\Pi^b_{k+1}$ then so are $(\forall x \leq t)A$ and $(\exists x \leq [t])A$.
   - (c) If $A, B \in \Pi^b_{k+1}$ then $A \land B$ and $A \lor B$ are in $\Pi^b_{k+1}$.
   - (d) If $A \in \Pi^b_{k+1}$ and $B \in \Sigma^b_{k+1}$ then $\neg A$ and $B \supset A$ are in $\Pi^b_{k+1}$.

4. $\Sigma^b_{k+1}$ and $\Pi^b_{k+1}$ are the smallest sets which satisfy (1)-(3).

This hierarchy of bounded formulae is in many respects analogous to the arithmetic hierarchy. The classes $\Sigma^b_k$ and $\Pi^b_k$ 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 $k \geq 1$. $\Sigma^b_k$ (respectively, $\Pi^b_k$) is the class of predicates which are defined by formulae in $\Sigma^b_k$ (respectively, $\Pi^b_k$).

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

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

\[ A \text{ bounded quantifier is one of the form } (\forall x \leq t) \text{ or } (\exists x \leq t) \text{ where } t \text{ can be any term. A sharply bounded quantifier is one of the form } (\forall x \leq [t]) \text{ or } (\exists x \leq [t]). \]
First we show $\Sigma^P_0$ contains all predicates defined by $\Sigma^P_1$ formulæ. All the nonlogical symbols of bounded arithmetic can be computed in polynomial time and hence are in $\Pi^P_1$. Since $\Sigma^P_0$ is closed under composition and since $\Sigma^P_1$ is closed under conjunction, disjunction and logarithmically bounded quantification, the desired result is established. The same argument also shows that $\Pi^P_1$ contains all predicates defined by $\Pi^P_1$ predicates.

For the reverse inclusion, let $R$ be an arbitrary predicate in $\Sigma^P_1$. By Theorem 7, $R$ can be written in the form

$$R(\mathcal{X}) = (\exists y \leq 2^{p(\mathcal{X})}) S(\mathcal{X}, y)$$

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 $\Sigma^P_1$-formula, it will suffice to show that $S$ is definable by a $\Sigma^P_1$-formula. To show that, we have to show that every occurrence of Choice, Truncate, $*$ and $\beta$ 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

$$F(b | a = 0) v (F(c) | a = 0).$$

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

Eliminating Truncate, $\beta$, and $*$ is a little more difficult. We shall show in great detail in Chapter 2 that $S$ is in fact equivalent to a $\Sigma^P_1$-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 $\Pi^P_k$ predicates are the negations of the $\Sigma^P_k$ predicates and the $\Pi^P_k$-formulae are equivalent to the negations of the $\Sigma^P_k$-formulae, we have immediately from the above that the $\Pi^P_k$ predicates are precisely the predicates definable by $\Pi^P_k$-formulae (when $k \geq 1$).

Q.E.D. □

1.7. Relativisation of the Polynomial Hierarchy.

The polynomial hierarchy can be relativised 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 \( \Omega \) is a function of polynomial growth rate whose domain is \( \mathbb{N}^k \) for some \( k \geq 1 \) and range is \( \mathbb{N} \). 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 \( i \):

1. \( \Omega_f(\Omega_1, \ldots, \Omega_k) = PTC(\Omega_1, \ldots, \Omega_k) \)
2. \( \Delta^f_2(\Omega_1, \ldots, \Omega_k) = PRED(\Delta^f_1(\Omega_1, \ldots, \Omega_k)) \)
3. \( \Sigma^f_2(\Omega_1, \ldots, \Omega_k) = PB\exists(\Delta^f_1(\Omega_1, \ldots, \Omega_k)) \)
4. \( \Pi^f_2(\Omega_1, \ldots, \Omega_k) = PB\forall(\Delta^f_1(\Omega_1, \ldots, \Omega_k)) \)
5. \( \Omega^f(\Omega_1, \ldots, \Omega_k) = PTC(\Sigma^f_1(\Omega_1, \ldots, \Omega_k)) \)
6. \( PH(\Omega_1, \ldots, \Omega_k) = \bigcup_f \Omega^f(\Omega_1, \ldots, \Omega_k) \)

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 \( \omega^f_j \) is equal to the set of all \( j \)-ary function oracles \( \Omega \) satisfying \( |\Omega(\vec{x})| \leq p(|\vec{x}|) \) for all \( \vec{x} \in \mathbb{N}^j \).

**Definition:** A functional \( f \) is a function with domain

\[ \mathbb{N}^{k_0} \times \omega^{f_1}_{k_1} \times \cdots \times \omega^{f_t}_{k_t} \]

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

The functional \( f \) has polynomial growth rate \( r(\vec{x}) \) if there is a suitable polynomial \( r(\vec{x}) \) such that for all \( \vec{r} \in \mathbb{N}^{k_0} \) and all function oracles \( \Omega_1, \ldots, \Omega_t \) with \( \Omega_j \in \omega^{f_j}_{k_j} \) for \( 1 \leq j \leq t \) we have

\[ |f(\vec{r}, \vec{\Omega})| \leq r(|\vec{r}|) \]

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

**Definition:** Let \( C \) be a set of functionals. Then \( PRED(C) \) is the set of members of \( C \) which have range \( \{0,1\} \).
\textbf{Definition:} Let $g$ and $h$ be functionals such that the domain of $g$ is 
\[
\mathbb{N}^k \times \omega_{n_1}^{r_1} \times \cdots \times \omega_{n_i}^{r_i}
\]
and the domain of $h$ is 
\[
\mathbb{N}^{k+2} \times \omega_{n_1}^{r_1} \times \cdots \times \omega_{n_i}^{r_i}
\]
Let $p$ and $q$ be $i$-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 $\tau$ be the functional with domain 
\[
\mathbb{N}^{k+1} \times \omega_{n_1}^{r_1} \times \cdots \times \omega_{n_i}^{r_i}
\]
so that for all oracles $\Omega_1, \ldots, \Omega_i$ with $\Omega_j \in \omega_{n_j}^{r_j}$ for $1 \leq j \leq i$ and for all $\bar{x} \in \mathbb{N}^k$, $\tau$ 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 $\bar{x}$, $n$ and $\Omega$ as above 
\[
|\tau(\bar{x}, n, \Omega)| \leq q(|\bar{x}|)
\]
and 
\[
f(x_1, \ldots, x_k, n, \Omega_1, \ldots, \Omega_i) = \tau(x_1, \ldots, x_k, f(|\bar{x}|), \Omega_1, \ldots, \Omega_i).
\]

\textbf{Definition:} Let $C$ be a set of functionals. We say that $C$ is \textit{uniform} iff there exists $\omega_{n_1}^{r_1}, \ldots, \omega_{n_i}^{r_i}$ such that every functional $f \in C$ has domain 
\[
\mathbb{N}^k \times \omega_{n_1}^{r_1} \times \cdots \times \omega_{n_i}^{r_i}
\]
for some $k_f$ which depends on $f$.

\textbf{Definition:} Let $C$ be a uniform set of functionals of polynomial growth rate. The domain of each functional in $C$ is of the form
The Polynomial Hierarchy

\[ \mathbb{N}^\Lambda \times \omega^{f_1}_n \times \cdots \times \omega^{f_k}_n \]

for some fixed \(\omega^{f_1}_n, \ldots, \omega^{f_k}_n\). The Polynomial-time closure, \(PTC(C)\), of \(C\) is the smallest uniform set of functionals containing \(C\) such that 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 \(x\) and all \(\Omega\),
   \[ g(x, \Omega) = f(x). \]
2. For each \(1 \leq j \leq i\), the functional \(P_j\) defined by
   \[ P_j(x_1, \ldots, x_n, \Omega_1, \ldots, \Omega_i) = \Omega_j(x_1, \ldots, x_n) \]
   is in \(C\).
3. \(C\) is closed under composition and under definition by limited iteration.

**Definition:** Let \(C\) be a set of functionals. Then \(PB^i(C)\) is the set of functionals \(Q\) such that \(Q\) has range \((0,1)\) and domain

\[ \mathbb{N}^{k+1} \times \omega^{f_1}_n \times \cdots \times \omega^{f_i}_n \]

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

\[ \mathbb{N}^{k+1} \times \omega^{f_1}_n \times \cdots \times \omega^{f_i}_n \]

such that for all \(\Omega_1, \ldots, \Omega_i\) with \(\Omega_j \omega^{f_i}_n\) for \(1 \leq j \leq i\), we have

\[ Q(x_1, \ldots, x_n, \Omega_1, \ldots, \Omega_i) = (\exists y \leq 2^{\mathbb{P}(\Omega)} R(x_1, \ldots, x_n, y, \Omega_1, \ldots, \Omega_i)) \]

\(PB^i(C)\) is defined similarly except that a bounded universal quantifier \((\forall y \leq 2^{\mathbb{P}(\Omega)}\) is used instead of the bounded existential quantifier.

We next define a polynomial hierarchy of functionals:

**Definition:** Let \(\omega^{f_1}_n, \ldots, \omega^{f_i}_n\) be a sequence of function oracles. The classes defined below are uniform sets of functionals which have domains of the form

\[ \mathbb{N}^\Lambda \times \omega^{f_1}_n \times \cdots \times \omega^{f_k}_n. \]
The definition is by induction on $j$:

1. $\Omega f(\omega_{n_1}^{r_1}, \ldots, \omega_{n_k}^{r_k}) = PTC(\emptyset)$
2. $\Delta f(\omega_{n_1}^{r_1}, \ldots, \omega_{n_k}^{r_k}) = PRED(\Omega f(\omega_{n_1}^{r_1}, \ldots, \omega_{n_k}^{r_k}))$
3. $\Sigma f(\omega_{n_1}^{r_1}, \ldots, \omega_{n_k}^{r_k}) = PB(\Delta f(\omega_{n_1}^{r_1}, \ldots, \omega_{n_k}^{r_k}))$
4. $\Pi f(\omega_{n_1}^{r_1}, \ldots, \omega_{n_k}^{r_k}) = PB(\Sigma f(\omega_{n_1}^{r_1}, \ldots, \omega_{n_k}^{r_k}))$
5. $\Omega f(\omega_{n_1}^{r_1}, \ldots, \omega_{n_k}^{r_k}) = PTC(\Sigma f(\omega_{n_1}^{r_1}, \ldots, \omega_{n_k}^{r_k}))$
6. $PH(\omega_{n_1}^{r_1}, \ldots, \omega_{n_k}^{r_k}) = \bigcup_j \Omega f(\omega_{n_1}^{r_1}, \ldots, \omega_{n_k}^{r_k})$

**Proposition 10:** Let $\omega_{n_1}^{r_1}, \ldots, \omega_{n_k}^{r_k}$ be a sequence of function oracles and let $\Omega_1, \ldots, \Omega_l$ be oracles so that $\Omega_j \succeq \omega_{n_j}^{r_j}$ for all $1 \leq j \leq l$. Let $k \geq 1$. Then for all functions $f$, $f \in \Omega f(\Omega_1, \ldots, \Omega_l)$ iff there exists a functional $g \in \Omega f(\omega_{n_1}^{r_1}, \ldots, \omega_{n_k}^{r_k})$ such that for all $x$,

$$f(x) = g(x, \Omega_1, \ldots, \Omega_l).$$

Similar statements hold for $\Delta f(\Omega_1, \ldots, \Omega_l)$, $\Sigma f(\Omega_1, \ldots, \Omega_l)$ and $\Pi f(\Omega_1, \ldots, \Omega_l)$.

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 $Bit: \mathbb{N} \to \mathbb{N}$ by limited iteration from $g_1$ and $h_1$ with bounds $p_1$ and $q_1$, where

$$g_1(i, x) = x$$

$$h_1(i, x, m, n) = Choice(m < i, \|y\|, y \in \{0\})$$

$$p_1(n, m) = m$$

$$q_1(n, m) = m$$

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 $f_2 \mathbb{N}^2 \rightarrow \mathbb{N}$ by limited iteration from $g_2$ and $h_2$ with bounds $p_2$ and $q_2$, where
\[
g_2(a,w) = 2 \cdot 2 \cdot w
\]
\[
h_2(a,w,m,v) = \text{Choice}(\text{Bit}(m,a,S(2(S(2v))))\text{,}2(S(2v)))
\]
\[
p_2(n,m) = n
\]
\[
q_2(n,m) = m + 2n + 2
\]
Set $a \ast w = \text{Choice}(a - 0.2(2(2 \cdot 2 \cdot w)), f_2(a, w))$.

(3) Define $f_3 \mathbb{N} \rightarrow \mathbb{N}$ by limited iteration from $g_3$ and $h_3$ with bounds $p_3$ and $q_3$, where
\[
g_3(w) = w
\]
\[
h_3(w,m,v) = \text{Choice}(4 \cdot \lfloor \frac{v}{4} \rfloor = v, \lfloor \frac{v}{4} \rfloor)
\]
\[
p_3(m) = m
\]
\[
q_3(m) = m
\]
Set $\text{Truncate}(w) = \lfloor g_3(w)/4 \rfloor$.

(4) Define $TR \ (i,w) : \mathbb{N}^2 \rightarrow \mathbb{N}$ by limited iteration from $g_4$ and $h_4$ with bounds $p_4$ and $q_4$, where
\[
g_4(i,w) = w
\]
\[
h_4(i,w,m,v) = \text{Choice}(Sm < i, \text{Truncate}(v), v)
\]
\[
p_4(n,m) = m
\]
\[
q_4(n,m) = m
\]
So $TR \ (i,w)$ is $\text{Truncate}$ applied $i-1$ times to $w$.

(5) Define $f_5 \mathbb{N}^2 \rightarrow \mathbb{N}$ by limited iteration from $g_5$ and $h_5$ with bounds $p_5$ and $q_5$, where
\[
g_5(i,w) = 0
\]
\[
h_5(i,w,m,v) = v v \{ - \text{Bit}(2m,w) v \sim \text{Bit}(S(2m),w) v m < i \}
\]
\[
p_5(n,m) = m
\]
\[
q_5(n,m) = 1
\]
So $f_5(i,w) = \begin{cases} 1 & \text{if } |\delta(i,w)| < i \\ 0 & \text{otherwise} \end{cases}$
(6) Define $f_6 : \mathbb{N} \rightarrow \mathbb{N}$ by limited iteration from $g_6$ and $h_6$ with bounds $p_6$ and $q_6$, where

\[
\begin{align*}
g_6(w) &= 0 \\
h_6(w,m,v) &= \text{Choice}(f_6(n,w), \text{Choice}(\text{Bit}(2m,w), S(2v), 2v), v) \\
p_6(n) &= n \\
q_6(n) &= n
\end{align*}
\]

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

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

\[
\begin{align*}
g_7(w) &= 0 \\
h_7(w,m,v) &= \text{Choice}((TR(m,w)) \sim TR(Sm,w), m, v) \\
p_7(n) &= n \\
q_7(n) &= n
\end{align*}
\]

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

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

\[
\beta(i,w) = \text{Choice}(i, f_6(TR(i,w)), f_7(w)).
\]

Q.E.D. $\Box$
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 ∨, ∧, ¬, →, =, 3, ∀ and parentheses and the nonlogical function symbols S, 0, +, ·, [x], [x]!, and # and the nonlogical predicate symbol ≤. 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, +, ·, and ≤ 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] = \lfloor \log_2(x+1) \rfloor. For example, [0] = 0. \lfloor x \rfloor denotes the greatest integer less than or equal to x/2. x#y is defined to be 2^{[x]!}.

We will frequently abbreviate x#y as xy. Also A\leftrightarrow B is an abbreviation for the formula (A\geq B)\rightarrow(B\geq 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#2=2 and \lfloor 2\rfloor=\lfloor 1\rfloor, we can use #, \lfloor x \rfloor, and \cdot to write the term 2^{[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^{[x]!} for some suitable polynomial p.

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

x #_{i+1} y = 2^{[x]!} #_i [x]!

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

28
Turing machines, we would use Turing machines with runtime bounded by terms involving \#. However we will not do this and we do not include \#n, \#k, ... in the language of Bounded Arithmetic.

Using 0, S, +, and \cdot we can construct terms to denote natural numbers. For example, both SSS0 and (S0)+S(0) are terms which denote the number 3. There are two canonical formats for terms which denote numbers. First, \(S^n0\) 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

\[
I_0 = 0 \\
I_{k+1} = I_k + (S0) \\
I_{2k+1} = (S0)(I_{k+1}) \\
\]

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 \(S^n0\). 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 \(\forall x \leq t\) or \(\exists x \leq t\) where \(t\) is any term not involving \(x\). A sharply bounded quantifier is a bounded quantifier of the form \(\forall x \leq |t|\) or \(\exists x \leq |t|\) 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. \(\Pi^b_0 = \Sigma^b_0 = \Delta^b_0\) is the set of formulae all of whose quantifiers are sharply bounded.

2. \(\Sigma^b_{k+1}\) is defined inductively by:

   a. \(\Sigma^b_{k+1} \supseteq \Pi^b_k\)

   b. If \(A\) is in \(\Sigma^b_{k+1}\) then so are \(\exists x \leq t \land A\) and \(\forall x \leq t \land A\).

   c. If \(A, B \in \Sigma^b_{k+1}\) then \(A \land B\) and \(A \lor B\) are in \(\Sigma^b_{k+1}\).
(4) If \( A \in \Sigma^b_{k+1} \) and \( B \in \Pi^b_{k+1} \) then \( \neg B \) and \( B \supset A \) are in \( \Sigma^b_{k+1} \).

(3) \( \Pi^b_{k+1} \) is defined inductively by:
(a) \( \Pi^b_{k+1} \supseteq \Sigma^b_{k+1} \).
(b) If \( A \) is in \( \Pi^b_{k+1} \) then so are \( (\forall x \leq t)A \) and \( (\exists x \leq t)A \).
(c) If \( A \) and \( B \) are in \( \Pi^b_{k+1} \) then \( A \land B \) and \( A \lor B \) are in \( \Pi^b_{k+1} \).
(d) If \( A \in \Pi^b_{k+1} \) and \( B \in \Sigma^b_{k+1} \) then \( \neg B \) and \( B \supset A \) are in \( \Pi^b_{k+1} \).

(4) \( \Sigma^b_{k+1} \) and \( \Pi^b_{k+1} \) are the smallest sets which satisfy (1)-(3).

Thus \( \Sigma^b_k \) and \( \Pi^b_k \) are defined analogously to the arithmetic hierarchy \( \Sigma_k^\emptyset \) and \( \Pi_k^\emptyset \) 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

\[
(\forall x \leq t)(\exists y \leq t)A(x,y) \iff (\exists w \leq (2t+1)\#(\text{id}(2t+1)))\exists z \leq t (A(x,\text{id}(z+1,w)) \land (z+1,w) \leq t)
\]

Essentially, \( w \) is a sequence which codes the values of \( y \) for each value of \( x \). We have not yet defined the \( \text{id} \) function in Bounded Arithmetic and obviously the quantifier bound for \( w \) depends on the precise definition of \( \beta \); 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 formul\(a_s \) 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 formul\(a_s \):

\[
\begin{align*}
(1) & \ y \leq x \land y \leq Sx \\
(2) & \ x \neq Sx \\
(3) & \ 0 \leq x \\
(4) & \ x \leq y \land x \neq y \rightarrow Sx \leq y \\
(5) & \ x \neq 0 \land x = 0 \\
(6) & \ y \leq x \land y \leq x \\
(7) & \ x \leq y \land x \leq x \\
(8) & \ x \leq y \leq z \land x \leq z \\
(9) & \ 0 = 0 \\
\end{align*}
\]
(10) $\langle 2, z = S(z) = S(\langle z, x z \rangle) = S(\langle z, x z \rangle)
(11) \langle S, z = S0, z = S \rangle$
(12) $\langle x, y = z = 2 \rangle$
(13) $\langle x, y = z = 2 \rangle$
(14) $\langle 0, y = z = S0, z = S \rangle$
(15) $\langle x, y = z = 2 \rangle$
(16) $\langle x, y = z = 2 \rangle$
(17) $\langle x, y = z = 2 \rangle$
(18) $\langle x, y = z = 2 \rangle$
(19) $\langle x, y = z = 2 \rangle$
(20) $\langle x, y = z = 2 \rangle$
(21) $\langle x, y = z = 2 \rangle$
(22) $\langle x, y = z = 2 \rangle$
(23) $\langle x, y = z = 2 \rangle$
(24) $\langle x, y = z = 2 \rangle$
(25) $\langle x, y = z = 2 \rangle$
(26) $\langle x, y = z = 2 \rangle$
(27) $\langle x, y = z = 2 \rangle$
(28) $\langle x, y = z = 2 \rangle$
(29) $\langle x, y = z = 2 \rangle$
(30) $\langle x, y = z = 2 \rangle$
(31) $\langle x, y = z = 2 \rangle$
(32) $\langle x, y = z = 2 \rangle$

(We are using 1 and 2 as abbreviations for the terms 0 and S0. Except for the results of Chapter 4, 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 $\Psi$ be a set of formulae. The $\Psi$-IND axioms are:

$$A(0) \land (\forall x)(A(x) \land A(S(x))) \Rightarrow (\forall x)A(x)$$

where $A$ is any formula in $\Psi$.

The $\Psi$-PIND axioms are:

$$A(0) \land (\forall x)(A(x) \land A(S(x))) \Rightarrow (\forall x)A(x)$$
where $A$ is any formula in $\Psi$.

The $\Psi$-$\text{IND}$ axioms are:

$$A(0)\land (\forall x)(A(x) \rightarrow A(Sx)) \lor (\forall x)A(|x|)$$

where $A$ is any formula in $\Psi$.

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

This is a good place to mention explicitly that we do not have the function $\text{z} \rightarrow 2^{\text{z}}$ in Bounded Arithmetic. Hence the conclusion $(\forall z) A(|z|)$ of $\Psi$-$\text{IND}$ is weaker than $(\forall z) A(z)$. Indeed, in a nonstandard model for Bounded Arithmetic the function $\text{z} \rightarrow 2^{\text{z}}$ may not be total and hence $\text{z} \rightarrow |\text{z}|$ 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) $\text{BASIC}$ axioms
   - (b) $\Sigma^0_i$-$\text{PIND}$ axioms.

2. $T_2^i$ has axioms:
   - (a) $\text{BASIC}$ axioms
   - (b) $\Sigma^0_i$-$\text{IND}$ axioms.

3. $S_2$ in $\bigcup S_2^i$.

4. $T_2$ is $\bigcup T_2^i$.

5. $S_2^{i+1}$ is the theory with only the $\text{BASIC}$ axioms. $T_2^{i+1}$ is the same theory.

Later we shall show that $T_2^{i+1} S_2^i$ and $S_2^{i+1} T_2^i$ 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 $k \geq 1$, $S_2^k$ 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: $\Psi-IND \implies \Psi-LIND$.

Proof: The hypotheses of $\Psi-IND$ and $\Psi-LIND$ are the same and the conclusion of $\Psi-IND$ is stronger than the conclusion of $\Psi-LIND$. □

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 formulæ in induction axioms.

Definition: Let $R$ be a fragment of Bounded Arithmetic. Suppose $A$ is a $\Sigma^b_i$-formula and that

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

and

$$R \vdash (\forall \bar{x})(\forall y)(\forall z)(A(\bar{x},y) \land A(\bar{x},z) \land y = z).$$

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

Definition: Let $f$ be a new function symbol. We define $\Delta^b_0(f), \Sigma^b_i(f)$ and $\Pi^b_i(f)$ to be sets of bounded formulæ in the language of Bounded Arithmetic plus the symbol $f$. These sets of formulæ are defined by counting alternations of bounded quantifiers, ignoring the sharply bounded quantifiers, exactly as in the definition of $\Delta^b_0, \Sigma^b_i$ and $\Pi^b_i$.

If $p$ is a new predicate symbol we define $\Delta^b_0(p), \Sigma^b_i(p)$ and $\Pi^b_i(p)$ similarly.

Theorem 2: Let $R$ be a fragment of Bounded Arithmetic. Suppose $R$ can $\Sigma^b_i$-define the function $f$. Let $B^* \in \Sigma^b_i$ be the theory obtained from $R$ by adding $f$ as a new function symbol and adding the defining axiom for $f$. Then, if $t > 0$ and $B$ is a $\Sigma^b_i(f)$- (or a $\Pi^b_i(f)$- ) formula, there is a $B^* \in \Sigma^b_i$ (or $\Pi^b_i$, respectively) such that $R \vdash B \iff B^* \vdash B$.

Proof: The defining axiom for $f$ is

$$f(\bar{x}) = y \iff A(\bar{x},y)$$

where $A$ is a $\Sigma^b_i$-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 $(Qz \leq s)B$ is a subformula of $B$ where the term $s$ involves $f$. Replace each occurrence of $f(t)$ in

$$\Delta^b_0(f), \Sigma^b_i(f), \Pi^b_i(f)$$

with $f$ as a new function symbol.
s by the term f(\(\varphi\)). (t is the bound in the \(\Sigma^k_1\)-definition of \(f\), see the definition above.) This yields a term \(s^*\). Now, \(\exists x \leq s^* D\) is provably equivalent to \((\exists x \leq s^*(x \leq s^* D))\) and \((\forall x \leq s^* D)\) is provably equivalent to \((\forall x \leq s^*)(x \leq s^* D)\). By repeating this procedure, we can form \(B_i\) so that

(1) \(R^* \vdash B_i \rightarrow B_1\), 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 \(R^* \vdash B_2 \rightarrow B_1\). Furthermore, if \(B\) is a \(\Sigma^k_1\) (or a \(\Pi^k_1\)) formula, then so are \(B_1\) and \(B_2\).

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

\[
B_2 = (Q_1 x_1 \leq s_1) \ldots (Q_n x_n \leq s_n) D
\]

where \(D\) is an open formula. Let \(f(\varphi)\) be a term appearing in \(D\). Obtain \(D^*\) by replacing \(f(\varphi)\) everywhere in \(D\) by a new variable \(x\). Define

\[
D_A = (\forall x \leq (f(\varphi))(A(x) \supset D^*)
\]

and

\[
D_B = (\exists x \leq (f(\varphi))(A(x) \land D^*).
\]

Let \(D^*\) and \(D^3\) be their respective prenex normal forms. Then \(D^*\) is a \(\Pi^k_1\) formula and \(D^3\) is a \(\Sigma^k_1\) formula, and

\[R^* \vdash (D \rightarrow D^*) \land (D \rightarrow D^3).\]

Define \(B_3\) from \(B_2\) by replacing the mantissa \(D\) by either \(D^*\) or \(D^3\), whichever is appropriate. We can do this so that \(B_3\) has the same alternation of (non-sharply) bounded quantifiers as \(B_2\). Also,

\[R^* \vdash B_3 \equiv B_2.\]

\(B_3\) was formed from \(B_2\) so that all occurrences of the term \(f(\varphi)\) 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. \(\Box\)
Corollary 3: Let $R$ be one of the theories $S^i_1$ or $T^i_2$ (where $i \geq 1$). Suppose $f_1, \ldots, f_k$ are functions $\Sigma^i_1$-definable by $R$. Let $\bar{R}$ be the theory obtained from $R$ by including new function symbols $f_1, \ldots, f_k$ and their defining axioms and including all $\Sigma^i_1(\bar{f})$-PIND axioms or $\Sigma^i_1(\bar{f})$-IND axioms (respectively). Then $\bar{R}$ 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 $\Sigma^i_1(\bar{f})$-formula is provably (in $R^*$) equivalent to a $\Sigma^i_1$-formula. Thus $R^*$-$\Sigma^i_1(\bar{f})$-PIND (or $\Sigma^i_1(\bar{f})$-IND respectively). Hence $\bar{R} = R^*$. □

The upshot of the last theorem is that we may freely adjoin $\Sigma^i_1$-definable functions to any fragment of Bounded Arithmetic and use these function symbols without restriction in induction formulas.

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 $\Sigma^i_1$ and $\Pi^i_1$ 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

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

($\bar{x}$ 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 $\Sigma^i_1(\bar{p})$- or $\Pi^i_1(\bar{p})$-formula then there is a $\Sigma^i_1$- or $\Pi^i_1$-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 $\Delta^i_1$ with respect to the theory $R$ if there are formulae $B \in \Sigma^i_1$ and $C \in \Pi^i_1$ 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 $\Delta^i_1$ when we mean $A$ is $\Delta^i_1$ with respect to $R$.

It follows immediately from Theorem 4 that if $A$ is a $\Delta^i_1$-formula, then a new predicate symbol $p$ can be introduced with the defining axiom $p(\bar{x}) \leftrightarrow A(\bar{x})$ and that $p$ can be used freely in formulæ 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 \( \cdot \) as

\[ x \cdot y = z \iff y + z = x \land x \leq y.\]

We show that \( \cdot \) can be \( \Sigma_1^b \)-defined in \( T_2^i \). To do this we have to show that \( T_2^i \) can prove

\[ (\forall z)(\exists y)(\exists x \leq z) M(x, y, z) \]

and

\[ M(x, y, z) \land M(x, y, z') \supset z = z'. \]

where \( M(x, y, z) \) is the formula on the right hand side of the defining axiom for \( \cdot \).

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 \( T_2^i \):

\[ (\exists z \leq 0) M(0, y, z) \]

\[ (\exists z \leq 0) M(x, y, z) \supset (\exists z \leq Sx) M(Sx, y, z). \]

From these two formulae we use \( \Sigma_1^b \)-IND to derive

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

Thus the subtraction function can be defined in \( T_2^i \).

We will later show by a much more complicated argument that the \( \cdot \) function can be \( \Sigma_1^b \)-defined in \( S_2^i \) as well.

As an application of the above example, we show that the theory \( T_2^i \) can derive the \( \Pi_1^b \)-IND axioms.

Theorem 5: The \( \Pi_1^b \)-IND axioms are theorems of \( T_2^i \) if \( i \geq 1 \)

Proof: Let \( A \) be a \( \Pi_1^b \)-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 \( \Sigma_1^b \)-formula \( \lnot A(y \cdot x) \). Then
Introducing Function and Predicate Symbols

\[ T_1^2 \vdash B(0, y) \land (\forall x) (B(x, y) \supset B(Sx, y)) \supset B(y, y) \]

or, equivalently,

\[ T_1^2 \vdash \neg \forall A(x) (\forall y) (\neg A(y \cdot x) \supset \neg A(y \cdot Sx)) \supset A(0). \]

From this we can readily derive

\[ T_1^2 \vdash A(0) \land (\forall x) (A(x) \supset A(Sx)) \supset A(y). \]

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 "walking on one's toes."

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 \( \Sigma^b_1 \)-defined and predicates be \( \Delta^b_1 \)-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 \( \Sigma_1^b \)-LIND axioms can be derived in \( S_2^1 \).

**Theorem 6:** Let \( i \geq 0 \). The \( \Sigma_i^b \)-LIND axioms are theorems of \( S_2^1 \).

**Proof:** Let \( A \) be a \( \Sigma_i^b \)-formula. We want to show that

\[ S_2^1 \vdash A(0) \land (\forall x) (A(x) \supset A(Sx)) \supset (\forall x) A(i[x]). \]
Let $B(x)$ be the formula $A([x])$. Then

$$S^2 \vdash A(0) \supset B(0)$$

and

$$S^2 \vdash (\forall x)(A(x) \supset A([x])) \supset (\forall x)(B([x]) \supset B(x)).$$

But $B$ is a $\Sigma^b_1$-formula so by the use of the $\Sigma^b_1$-$\text{PIND}$ axiom for $B$ we get

$$S^2 \vdash A(0) \cup (\forall x)(A(x) \supset A([x])) \supset (\forall x)B(x)$$

which is what we wanted to show.

Q.E.D. □

We bootstrap $S^2$ by showing that the following functions and predicates are $\Sigma^b_1$-definable in $S^2$ and are $\Delta^b_1$ with respect to $S^2$, respectively.

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

$$a < b \iff a \leq b \land a \neq b$$

$$c = \max(a, b) \iff (c \geq a \land c = b) \lor (c \geq b \land c = a)$$

$$c = \min(a, b) \iff (c \leq a \land c = b) \lor (c \leq b \land c = a)$$

The uniqueness and existence conditions for these functions follow easily from the $\text{BASIC}$ axioms without any use of induction. Since the defining formula for $a < b$ is open it is trivially $\Delta^b_1$.

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

$$b = P(a) \iff (a = 0 \land b = 0) \lor Sb = a.$$

The uniqueness condition for this definition follows easily from the $\text{BASIC}$ 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

$$(\exists x < 0)M(0, x)$$

and

$$(\exists x \leq [x]_{\mathbb{Z}})M([x]_{\mathbb{Z}}, x) \supset (\exists x \leq x)M(x, x)$$

from the $\text{BASIC}$ axioms again without any use of induction axioms. Finally, $\Sigma^b_1$-$\text{PIND}$
yields

\[ S_2^1 \vdash (\forall x)(\exists z \leq x)M(x, z). \]

(c) \textit{Power}_2(a) \iff S[|a|] = |a|

This predicate symbol is clearly \( \Delta_1 \) with respect to \( S_2^1 \). Moreover, \( S_2^1 \) can prove many nice properties of \textit{Power}_2. In particular, \( S_2^1 \) can prove the following formulae:

\[
\begin{align*}
\textit{Power}_2(a) & \supset \lnot \exists z \\
\textit{Power}_2(a) & \supset \textit{Power}_2(a + a) \\
\textit{Power}_2(a) & \supset \textit{Power}_2(\lfloor a \rfloor) \\
\textit{Power}_2(a) & \supset \textit{Power}_2(\lfloor a \rfloor) \supset \lnot \exists z \supset \lnot \exists z = a - b \\
\textit{Power}_2(a) & \supset \textit{Power}_2(\lfloor a \rfloor) 
\end{align*}
\]

For example, the fourth formula follows from the open formula

\[ a - \textit{S}(c) \land a - \textit{S}(d) \supset \textit{S}(\lfloor c \rfloor) \land b - \textit{S}(d) \supset \textit{S}(\lfloor d \rfloor) \land a - \textit{S}(d) \supset a - b \]

which in turn can be proved in \( S_2^1 \) (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 \( S_2^1 \).

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

\[ e = \text{Exp}(a, b) \iff \text{Power}_2(\lfloor c \rfloor) \land |c| = 1 + \min(|b|, a) \]

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

Let \( M(a, b, c) \) be the formula on the right-hand side of the definition of \textit{Exp}. Then, by the properties of \textit{Power}_2 discussed above,

\[ S_2^1 \vdash M(x, y, z) \land M(x, y, u) \supset z = u. \]

Also,

\[
\begin{align*}
S_2^1 & \vdash M(x, y, z) \land x \leq |y| \supset M(Sz, y, 2z) \\
S_2^1 & \vdash M(x, y, z) \land x \geq |y| \supset M(Sy, z) \\
S_2^1 & \vdash M(x, y, z) \land z \leq 2y + 1
\end{align*}
\]

From this we get
\[ S_2^1 \vdash (\exists z \leq 2y+1)M(x,y,z) \supset (\exists z \leq 2y+1)M(Sx,y,z). \]

Hence,
\[ S_2^1 \vdash (\forall x \leq y)(\exists z \leq 2y+1)(x \leq u \supset M(x,y,z)) \supset (\forall x \leq y)(\exists z \leq 2y+1)(x \leq Su \supset M(x,y,z)). \]

On the other hand, \( S_2^1 \vdash (\forall y)M(0,y,1) \) so by \( \Sigma_1^L - \text{LIND} \) with respect to the variable \( y \) (remember, we don’t count sharply bounded quantifiers):
\[ S_2^1 \vdash (\forall x \leq y)(\exists z \leq 2y+1)(x \leq y \supset M(x,y,z)) \]
and hence
\[ S_2^1 \vdash (\forall x \leq y)(\exists z \leq 2y+1)M(x,y,z). \]

And since \( S_2^1 \vdash x \geq y \supset M(x,y,z) \supset M(x,y,z) \), we have
\[ S_2^1 \vdash (\forall x)(\exists z \leq 2y+1)M(x,y,z). \]

This is what we needed to demonstrate that \( \text{Exp} \) is properly defined in \( S_2^1 \).

It is important to note that we have not defined exponentiation, but only a restricted exponentization. Indeed, in the formula
\[ x - \text{Exp}(i,y) = 2^{\min(i,y)} \]
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 \( (\forall i \leq z)[z \mid |2^i|] \) instead of the more correct \( (\forall i \leq z)[z \mid 2^{\min(i,z)}] \).

(e) \( b = \text{Mod}(a) \iff b \equiv 2 \cdot \lfloor \frac{1}{2}a \rfloor \mod a \)

\( \text{Mod}(a) \) 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 \text{BASIC} axioms.

(f) We define functions for obtaining the “less significant part” and the “more significant part” by defining the following predicate and functions:
\[ \text{Decomp}(a,b,c,d) \iff \exists e \leq b \cdot 2^{\min(d,b)} c = e \]
\[ e = \text{LSP}(a,b) \iff (\exists c \leq a) \text{Decomp}(a,b,c,d) \]
\[ d = \text{MSP}(a,b) \iff (\exists c \leq a) \text{Decomp}(a,b,c,d) \]
Clearly, $\text{Decomp}$ is $\Delta^1_1$-defined. Also, it is not difficult to see that

$$S_2^1 \vdash \text{Decomp}(a,b,c,d) \land \text{Decomp}(a,b,c,d') \iff e = a, d = f.$$

This establishes the uniqueness conditions for both MSP and LSP.

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

$$S_2^1 \vdash (\exists e \leq a)(\exists a \leq n)\text{Decomp}(a,b,c,d).$$

Since $S_2^1 \vdash \text{Decomp}(0,0,0,0)$, we know that

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

Also, the following are provable in $S_2^1$:

$$x < |a| \land \text{Decomp}(a,x,c,d) \supset \text{Decomp}(a,Sx,c + 2^x, \text{Mod}(d, \lfloor \frac{1}{2} d \rfloor))$$

and

$$\text{Decomp}(a,b,c,d) \supset c \leq a \land d \leq a.$$ 

It follows readily that

$$S_2^1 \vdash (\forall x \leq |a|)(\exists e \leq a)(\exists d \leq a)(x \leq a \supset \text{Decomp}(a,x,1, 1, d)) \supset$$

$$\supset (\forall x \leq |a|)(\exists e \leq a)(\exists d \leq a)(x \leq a \supset \text{Decomp}(a,x,c,d)).$$

From this, by use of $\Sigma^1_3$-LIND

$$S_2^1 \vdash (\forall x \leq |a|)(\exists e \leq a)(\exists d \leq a)\text{Decomp}(a,x,c,d),$$

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

$$e = \text{Bit}(b,a) \iff e = \text{Mod}(\text{MSP}(a,b)).$$

So $\text{Bit}(b,a)$ is the value of the bit in the $2^a$ position of the binary representation of $a$. Since $\text{Bit}$ is defined as the composition of functions already introduced in $S_2^1$, it is clear that $\text{Bit}$ is $\Sigma^1_3$-defined.

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

$$|a| \geq |b| \land (\forall y < |a|)(\text{Bit}(y,a) = \text{Bit}(y,b)) \iff a = b.$$ 

That is, it is provable that the binary representation of a number uniquely determines
the number. This can be proved in $S^1_2$ by using $\Sigma^1_2$-LIND with respect to the variable $u$ on the formula

$$(\forall y < u) [\text{Bit}(y, a) = \text{Bit}(y, b)] \Rightarrow \text{LSP}(a, u) \Rightarrow \text{LSP}(b, u).$$

The details are left to the reader.

Further note that $S^1_2$ can prove all the simple relationships between Bit, MSP and LSP; for example,

$S^1_2 \vdash \text{Bit}(b, a) = \text{MSP}(\text{LSP}(a, 0), b)$

$S^1_2 \vdash \text{Bit}(b, a) = \text{LSP}(\text{MSP}(a, b), 1)$

$S^1_2 \vdash \text{Bit}(b + c, a) = \text{Bit}(c, \text{MSP}(a, b)).$

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

$$c = \text{LENMINUS}(a, b) \iff (b \leq a \land c + b = a) \lor (b \geq a \land c = 0).$$

So $\text{LENMINUS}(a, b)$ is equal to $[a] + b$, or in other words, $\text{LENMINUS}$ 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 $\Delta^0_1$-formulas. It will suffice to show that

$S^1_2 \vdash (\forall x \leq |a|)(\exists y \leq |a|)[x + y = |a|].$

Now, $S^1_2 \vdash x < |a| \land x + y = |a| \Rightarrow S(x) + P(y) = |a|.$

So,

$S^1_2 \vdash (\forall x \leq |a|)(\exists y \leq |a|)[x + y = |a|], (\forall x \leq |a|)(\exists y \leq |a|)[x \leq S(x) + y = |a|].$

By $\Sigma^1_1$-LIND we obtain the desired result.

(i) Finally, we show that subtraction can be $\Sigma^1_1$-defined in $S^1_2$.

$$c = a - b \iff a + b = c \land (c = 0 \land a < b)$$

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 $\Sigma^1_1$-LIND on the formula $\text{M}(a, b, c)$ defined as
\[ b \leq a \supset (\exists x \leq a)(x + \text{MSP}(b, a - x) = \text{MSP}(a, x - a_0)). \]

Here we are using \([a] - u\) as an abbreviation for \(\text{LENMINV}(a, u)\). Now,

\[ S_1^+ [b \leq a] : \text{MSP}(b, [a]) = 0. \]

So \( S_1^+ : M(a, b, 0) \). Also, \( S_1^+ : M(a, b, u) \supset M(a, b, S_0 u) \) can be proved without too much difficulty; this follows from the fact that \( S_1^+ \) can prove all of the following:

(i) \( 1 \leq a \supset \text{MSP}(b, x) < \text{MSP}(a, x) \supset (\text{MSP}(b, x) = \text{MSP}(a, x) \supset \text{LSP}(b, x) \leq \text{LSP}(a, x)) \)

(ii) \( b \leq a x + \text{MSP}(b, S_0 x) \supset \text{MSP}(a, S_0 x) \supset \text{Bit}(x, a) \supset \text{Bit}(x, b) \supset (2x + 1) \supset \text{MSP}(b, x) = \text{MSP}(a, x) \)

(iii) \( b \leq a x + \text{MSP}(b, S_1 x) \supset \text{MSP}(a, S_1 x) \supset \text{Bit}(x, a) \supset \text{Bit}(x, b) \supset (2x + 1 + \text{MSP}(b, x) = \text{MSP}(a, x) \)

(iv) \( b \leq a x + \text{MSP}(b, S_2 x) \supset \text{MSP}(a, S_2 x) \supset \text{Bit}(x, a) \supset \text{Bit}(x, b) \supset (2x + 1 + \text{MSP}(b, x) = \text{MSP}(a, x) \)

By \( \Sigma_1^+ \mid \text{LIN} \), \( S_1^+ : M(a, b, [a]) \), which is equivalent to the existence condition for the definition of subtraction since \( S_1^+ : \text{MSP}(x, 0) = x \).

(j) \( \text{QuoRem}(a, b, c, d) \iff (b = 0 \land c = 0 \land d = 0) \lor (d < b \land a = c - b + d) \)

\( c = [a/b] \iff (\exists b) \text{QuoRem}(a, b, c, d) \)

\( d = \text{Rem}(a, b) \iff (\exists c < a) \text{QuoRem}(a, b, c, d) \)

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 \( \frac{a}{b} \) ?)

(k) \( b \mid a \iff \text{Rem}(a, b) = 0, b \neq 0 \)

(l) \( \text{Even}(a) \iff \text{Mod2}(a) = 0 \)

\( \text{Odd}(a) \iff \text{Mod2}(a) = 1 \)
\( \text{Comma}(b, a) \iff \text{Even}(b) \land \text{Bit}(b, a) = 1 \land \text{Bit}(Sb, a) = 0 \)

\( c = \text{Digit}(b, a) \iff [\text{Even}(b) \land \text{Bit}(Sb, a) = 1 \land \text{Bit}(b, a) = c] \lor [\text{Odd}(b) \land \text{Bit}(Sb, a) = 0 \land c = 2] \)

\( \text{Comma} \) and \( \text{Digit} \) are immediately seen to be \( \Delta^b \)-definable and \( \Sigma^b_1 \)-definable by \( S^b_2 \). 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.

\( \text{PSqSl}(b, c) \iff |a|+2 = 2c \land Sb, l(y < |a|) ((\exists b (b+2) \land (y+2) > \text{Comma}(y, a)) \land \forall (y < |a|) \lnot (\text{Even}(y) \land (2y+2) \land (y+2) > \text{Digit}(y, a)) \neq 2) \)

\( b = \text{ProtoSize}(a) \iff 2 \leq |a| \land (2b = |a| \lor \text{Comma}(2 \cdot b, a) \lor (\forall y < b) (\lnot \text{Comma}(2 \cdot x, y)) \)

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

\( \text{ProtoSeq}(a) \iff \text{PSqSl}(a, \text{ProtoSize}(a), \text{ProtoLen}(a)) \)

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. \( \text{PSqSl}(a, b, c) \) 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 \( \text{ProtoLen} \) and \( \text{ProtoSize} \) can be \( \Sigma^b_1 \)-definable in \( S^b_1 \) and that \( \text{PSqSl} \) and \( \text{ProtoSeq} \) are \( \Delta^b \) with respect to \( S^b_2 \).

\( c = \text{Proto}(b, a) \iff \lnot [\text{ProtoSeq}(a) \land c = 0] \lor [\text{ProtoSeq}(a) \land c \leq \text{ProtoSize}(a) \land \forall (y < \text{ProtoSize}(a)) [\text{Bit}(b, y) = \text{Digit}(2 \cdot y + (\text{ProtoSize}(a) + 1) \cdot (b - 1), a)] \)

So if \( a = a_1, \ldots, a_k \) then \( \text{Proto}(a, i) = i \). 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 \( a_k \) coded by the high order bits.

The uniqueness condition for \( \text{Proto} \) 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 \( S^b_2 \).

It is important to note that since \( S^b_1 \)-\( \text{ProtoSize}(a) \leq |a| \), the quantifier \( \forall y < \text{ProtoSize}(a) \) can be replaced by a sharply bounded quantifier. This makes it possible to prove the existence condition for \( \text{Proto} \) by using \( \Sigma^b_2 \)-LIIND with respect to the variable \( a \) on the formula.
$\text{ProtoSeq}(a) \land u < \text{ProtoSize}(a) \supseteq$
\[ \exists (3 \leq a \mid |e| \leq w \mid \forall y \leq a \mid \text{Bit}(y, c) = \text{Digit}(2 \cdot (y + (\text{ProtoSize}(a)+1) \cdot (b-1)), a)) \].

$S^2_2$ also proves that protosequences exist. Indeed, it can be shown by induction on the length of $a$ that

$S^2_2 \vdash \exists b \leq |d| \mid \exists a \leq b \vdash \exists x \leq 4 \cdot d^2 \cdot \text{PSQL}(x, \beta, 1) \land \text{Proto} \beta[1, x] = a.$

(p) $c = \text{ProtoStar}(a, b) \iff \neg \text{ProtoSeq}(a) \land c = 0 \lor \exists y \text{ProtoSeq}(c) \land \text{ProtoSeq}(a) \land \forall \text{ProtoSize}(c) = \text{ProtoSize}(a) \land \text{ProtoLen}(c) = 1 \land \text{ProtoLen}(a) \land \forall \text{Proto}(\text{ProtoLen}(a)+1, c) = \text{LSP}(b, \text{ProtoSize}(a)).$

$\text{ProtoStar}(a, b)$ 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 $\text{ProtoStar}$.

The reader may supply them if desired.

2.5. Bootstrapping $S^2_2$ - Phase 2.

For the second phase of bootstrapping $S^2_2$ 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:

- $\text{Seq}(w)$ true iff $w$ is a valid sequence
- $\text{Size}(w)$ the maximum of the lengths of entries of $w$
- $\text{Len}(w)$ 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 $\text{Seq}$, $\text{Size}$, * and ** since each of these can be defined by "local" operations. However, $\text{Len}(w)$ and $\beta$ are harder to define. Computing $\text{Len}(w)$ 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 \( A(x,y,z) \) be any formula. The function \( f(y,\bar{x}) \) is defined by length bounded counting from \( A \) iff \( f \) satisfies

\[
f(y,\bar{x}) = (\#z \leq |y|)A(x,y,z)
\]

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

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 \( A(x,y,z) \) be \( \Delta^1_1 \) with respect to \( S^1_2 \). Let \( f \) be the function defined by length bounded counting from \( A \). Then \( f \) can be \( \Sigma^1_1 \)-defined in \( S^1_2 \).

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

\[
c = g(x,\bar{x}) \iff |c| \leq |x| + \sum_{k} |y| \leq |x| + |x| = 1 \Rightarrow A(x,y,\bar{x}).
\]

The existence and uniqueness conditions from \( g \) are readily proved in \( S^1_2 \). For the existence condition we use \( \Sigma^1_1 \)-LIND with respect to \( u \) on the formula

\[
(\exists c \leq t)(\forall x \leq |x| + 1)[B(x,c) = 1 \Rightarrow A(x,b,\bar{x})].
\]

Note that we needed the fact that \( A \) is \( \Delta^1_1 \) in order to \( \Sigma^1_1 \)-define \( g \).

We define the function \( \text{Numoness} \), which computes the number of ones in the binary representation of a number \( a \), by

\[
b = \text{Numoness}(a) \iff (\exists w \leq 2^{2^n+1})(\forall i<n)[P_{\text{QSL}}(\bar{w},|a|,|a|+1)\land
\]

\( n \text{Proto}(i+1,w) = 0 \land n = \text{Proto}(i+1,0,w) \land
\]

\( \forall i<n)[P_{\text{Proto}}(i+1,w) = 0 \land P_{\text{Proto}}(i+1,w) + B(i,a) = P_{\text{Proto}}(i+2,w)].
\]

The uniqueness and existence conditions for \( \text{Numoness} \) are provable in \( S^1_2 \) 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 \( \text{Numoness} \); this is the first time we have used the \( \# \)
function for bootstrapping \( S^1_2 \).

We can now define \( f \) as

\[
f(b, \bar{x}) = \text{Numones}(g(b, \bar{x})).
\]

Q.E.D. \( \square \)

**Theorem 8:** \( S^1_2 \) proves the following:

\[
\text{Numones}(x) + \text{Numones}(y) = \text{Numones}(x + y + y).
\]

**Proof:** It is not hard to prove this using \( \Sigma^1_1 \)-LIND. This depends on the fact that \( S^1_2 \) can prove simple properties about \( Bm \), \( MSP \) and \( LSP \); see \( \S 2.4(\text{g}) \). \( \square \)

**Theorem 9:** The following functions are \( \Sigma^1_1 \)-definable in \( S^1_2 \) and hence can be introduced as defined function symbols.

(i) \( f_1(\bar{x}) = \min\{t(y) \mid y \leq |s|\} \)

(ii) \( f_2(\bar{x}) = \max\{t(y) \mid y \leq |t|\} \)

(iii) \( f_3(\bar{x}) = (\exists y \leq |s|[A(y)]) \)

where \( s \) and \( t \) are terms and \( A \) is a \( \Delta^1_1 \)-formula. The free variables of \( s \) are the \( \bar{x} \); the free variables of \( t \) and of \( A \) may include \( y \) and \( \bar{x} \). The terms \( s \) and \( t \) may involve \( \Sigma^1_1 \)-defined function symbols.

**Proof:** The existence and uniqueness conditions for \( f_1 \) and \( f_2 \) can be proved easily by using \( \Sigma^1_1 \)-LIND on the length of \( s \). We can define \( f_3 \) in terms of \( \text{Numones} \) by

\[
f_3(\bar{x}) = (\exists y \leq |s|)(\forall z \leq |s|)(z \leq y \rightarrow A(z)).
\]

\( \square \)

**Lemma 10:** Let \( f_1, f_2, \) and \( f_3 \) be the function symbols introduced in Theorem 9. Then \( S^1_2 \) proves the following:
(i) \( \exists y \leq |x| \exists f(y)(\overline{f(y,x)} \leq \ell(y,x)) \land (\forall y \leq |x|)(\ell(f(y,x)) \leq \ell(y,x)) \)

(ii) \( \exists y \leq |x| \exists f(y)(\overline{f(y,x)} \leq \ell(y,x)) \land (\forall y \leq |x|)(\ell(\overline{f(y,x)}) \geq \ell(y,x)) \)

(iii) \( (\forall x < f(y,x))(\neg A(y)) \land (\exists y \leq |x|)(A(\overline{f(y,x)})) \land (\ell(\overline{f(y,x)}) \leq |y| + 1) \)

**Proof:** This is actually what we proved in Theorem 9. Note we have defined \( f_3 \) so that if \( (\forall y \leq |x|)A(y) \) then \( f_3(y,x) = |x| + 1. \)

We are now ready to introduce function and predicate symbols in \( S^2 \) for handling general sequences. We leave the provability of the necessary uniqueness and existence conditions to the reader.

(a) \( b = \text{Substring}(a, i, j) \iff b = \text{MSP}(LSP(a, j), i) \)

So the binary representation of \( b \) is that portion of \( a \)'s binary representation starting with the \( 2^{i-1} \) bit and ending with the \( 2^j \) bit. For example, if \( a = 11010 \) then \( \text{Substring}(a, 0, 3) = 101 = 5_{10} \) and \( \text{Substring}(a, 1, 3) = 110 = 6_{10} \).

(b) \( \text{Seq}(w) \iff (\forall x < |w|)(\text{Even}(i) \lor \text{Comma}(i, w) \lor \text{Digit}(i, w) \leq 1) \land (\text{Comma}(0, w) \lor w = 0) \)

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) \( x = \text{Len}(w) \iff (\neg \text{Seq}(w) \lor a = 0) \lor (\text{Seq}(w) \land a = (\# < |w|) \land \text{Comma}(i, w)) \)

(d) \( b = \text{Decode}(a) \iff (b = 0) \lor \text{ProtoSeq}(a)(b = \text{Proto}(1, a)) \)

\( b = \text{Encode}(a) \iff \text{FSqL}(b, |w|, 1) \land a = \text{Proto}(1, b) \)

The existence condition for \( \text{Encode} \) follows from the remark made in \([2,4(c)]\) above.
(e) $a = \text{Start}(i, w)$ $\iff$ $(a = 0 \land \neg \text{Seq}(w)) \lor (\text{Seq}(w) \land \\
\land a = (\mu z \leq |w| + 1) [\text{Len}(\text{Substring}(w, 0, z)) = i, \text{Even}(z)])$

$b = \text{End}(i, w)$ $\iff$ $(a = 0 \land \neg \text{Seq}(w)) \lor (\text{Seq}(w) \land \\
\land b = (\mu z \leq |w|) [\text{Len}(\text{Substring}(w, \text{Start}(i, w), z + 2)) \neq 0, \text{Even}(z)])$

$a = \beta(i, w)$ $\iff$ $(i = 0 \land a = \text{Len}(w)) \lor \\
\lor (a = \text{Decode}(\text{Substring}(w, \text{Start}(i, w), \text{End}(i, w))) \land i \neq 0)$

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

(f) $\text{Sinc}(w) = \max \{ [\text{End}(i, w) \land \text{Start}(i, w)] : i \leq \text{Len}(w) \}$

(g) $a * b = b * (a + b)$

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

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

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

(j) $\text{UniqSeq}(w) \iff \text{Seq}(w) \land \forall w. \exists w. w \neq 0 \land \\
\land \forall z \leq |w| [\text{Digit}(i, w) = 0 \land \text{Comma}(i + 2, w)]$

$\text{UniqSeq}(w)$ 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 $\text{UniqSeq}$ is that $S_2^\text{p}$ proves

$\text{UniqSeq}(a) \land \text{UniqSeq}(b) \land (\forall i \leq \text{Len}(i)[\beta(i, a) = \beta(i, b)]) \Rightarrow a = b$

and

$\text{Seq}(a) \Rightarrow (\exists w [\text{UniqSeq}(w) \land (\forall i \leq \text{Len}(i)[\beta(i, a) = \beta(i, w)]))$.}

(k) $\text{Seq}(a, b) = (2b + 1) \#(4(2a + 1)^2)$
$SqBd$ is useful since

$$S^2_1 \vdash \text{UniqSeg}(w) \iff \lambda x. \text{Len}(w) \leq |\beta| + 1 \forall i < \text{Lex}(w) \exists \beta \exists S \forall i \forall x \forall y \exists z \exists w < SqBd(a, b).$$

### 2.6. Bootstrapping $T^1_2$.

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 $\Sigma^1_2$--$\text{PIND}$ axioms. Hence $T^1_2 \equiv S^2_1$ and all the functions defined in the last two sections can be introduced into $T^1_2$.

We begin by showing that the following functions may be introduced in $T^1_2$:

(a) \[
  a < b \iff a \leq b \land a \neq b
\]

\[
  c = \max(a, b) \iff (c \geq a \land c = b) \lor (c \geq b \land c = a)
\]

\[
  c = \min(a, b) \iff (c \leq a \land c = b) \lor (c \leq b \land c = a)
\]

(b) \[
  b = P(a) \iff (a = 0 \land b = 0) \lor b = a
\]

We showed in an example earlier that subtraction is $\Sigma^1_2$--definable in $T^2_1$. So we can define $P(a) = a + 1$.

(c) \[
  \text{Power}^2(a) \iff S(|P(a)|) = |a|
\]

When we introduced $\text{Power}^2$ as a defined predicate symbol of $S^2_1$ we showed that $S^2_1$ can prove many basic properties of the $\text{Power}^2$ predicate. The same comments apply to $T^1_2$.

(d) \[
  c = \text{Exp}(a, b) \iff \text{Power}^2(c)|c| = 1 + \text{min}(|b|, a)
\]

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

Let $M(a, b, c)$ be the right-hand side of the defining equation for $\text{Exp}$. Then,

$$T^1_2 \vdash M(x, y, z) \iff (x \leq |y| \land |S_x| = |y|) \implies M(x, S_y, z)$$

and

$$T^1_2 \vdash M(x, y, z) \iff x > |y| \land |S_y| > |y| \implies M(x, S_y, 2z).$$

Hence,
\[ T_2^1 \vdash (\exists x \leq 2y + 1)M(x, y, z) \Rightarrow (\exists z \leq 2\cdot 5^{y+1})M(x, S^y, z). \]

Since \( T_2^1 \vdash M(x, 0, 1) \), we can use \( \Sigma^1_2 \)-IND to obtain
\[ T_2^1 \vdash (\forall z)(\exists y)(\exists x \leq 2y + 1)M(x, y, z) \]

which is the existence condition for the definition of \( \text{Exp} \).

\( \text{(e) } \text{Decomp}(a, b, c, d) \iff |b| \leq h(d, 2^{\min(|a|, 1)}), c = a \)

\[ c = \text{LSP}(a, b) \iff (\exists d \leq a) \text{Decomp}(a, b, c, d) \]

\[ d = \text{MSP}(a, b) \iff (\exists e \leq a) \text{Decomp}(a, b, c, d) \]

It is easy to see that \( \text{Decomp} \) may be introduced as a \( \Delta^1_1 \)-defined predicate symbol of \( T_2^1 \). Also, the uniqueness conditions for \( \text{LSP} \) and \( \text{MSP} \) follow from the \( \text{BASIC} \) axioms. It will suffice therefore to show that
\[ T_2^1 \vdash (\exists x \leq a)(\exists y \leq x) \text{Decomp}(a, b, x, y). \]

But
\[ T_2^1 \vdash \text{Decomp}(a, b, c, d) \land x + 1 < 2^{\min(|a|, 1)} \text{Decomp}(a + 1, b, c + 1, d) \]

and
\[ T_2^1 \vdash \text{Decomp}(a, b, c, d) \land x + 1 \geq 2^{\min(|a|, 1)} \text{Decomp}(a + 1, b, 0, d + 1). \]

Hence we can use \( \Sigma^1_2 \)-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 \( \Gamma \geq 1 \). \( T_2^1 \) proves the \( \Sigma^1_2 \)-PIND axioms. Hence \( T_2^1 \vdash \Sigma^1_2 \).

**Proof:** Let \( A \) be any \( \Sigma^1_2 \)-formula. We want to show that
\[ T_2^1 \vdash A(0) \land (\forall z)(A(\text{L}_z) \Rightarrow A(z)) \Rightarrow A(z) \]

(where \( a \) is a free variable which appears only as indicated.) Let \( B(a, u) \) be the formula
\[ A(\text{MSP}(a, a) \land u) \]. Then
\[ T^2_2 \vdash A(0) \supset B(a,0) \]
\[ T^2_2 \vdash (\forall x)(A(\lfloor x \rfloor) \supset A(x)) \supset (\forall x)(B(x,x) \supset B(a,Sx)). \]

Now, from \( \Sigma^1_1 \text{-LIND} \) on \( B \), we have
\[ T^2_2 \vdash B(\lfloor x \rfloor)(B(x) \supset B(Sx)) \supset B(\lfloor \lfloor a \rfloor \rfloor)). \]

But,
\[ T^2_2 \vdash B(\lfloor a \rfloor) \supset A(\lfloor a \rfloor). \]

Q.E.D. □

If we examine the proof of Theorem 11, we note that only \( \Sigma^1_1 \text{-LIND} \) is used, not \( \Pi^1_1 \text{-IND} \). Hence what we have proved is:

**Theorem 13:** Let \( R_i \) be the theory \( S^2_i \) plus the \( \Sigma^1_1 \text{-LIND} \) 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 1 implies that \( S^2_i \vdash R_i \).

Q.E.D. □

**Theorem 15:** Let \( i \geq 0 \). Then
(a) \( S^2_i + \Sigma^1_1 \text{-LIND} \) is equivalent to \( S^2_i + \Sigma^1_1 \text{-PIND} \).
(b) \( S^2_i + \Sigma^1_1 \text{-IND} \) is equivalent to \( S^2_i + \Pi^1_1 \text{-IND} \).
(c) \( S^2_i + \Sigma^1_1 \text{-LIND} \) is equivalent to \( S^2_i + \Pi^1_1 \text{-LIND} \).
(d) \( S^2_i + \Sigma^1_1 \text{-PIND} \) is equivalent to \( S^2_i + \Pi^1_1 \text{-PIND} \).

**Proof:** The inclusion of \( S^2_i \) means that we can use all of the \( \Sigma^1_1 \)-defined function symbols of \( S^2_i \) 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 \( S^2_i + \Pi^1_1 \text{-LIND} \) is equivalent to \( S^2_i + \Pi^1_1 \text{-PIND} \).

\[ S^2_i + \Pi^1_1 \text{-PIND} \Rightarrow \Pi^1_1 \text{-LIND} \]

follows from the proof of Theorem 6 modified so that \( A \in \Pi^1_1 \) instead of \( \Sigma^1_3 \). Likewise, the proof of Theorem 11 modified so that \( A \in \Pi^1_1 \) shows that \( S^2_i + \Pi^1_1 \text{-LIND} \Rightarrow \Pi^1_1 \text{-PIND} \).

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 \((\forall x \leq t)(\exists y \leq s)A(x,y) \iff (\exists t)(\forall x \leq t)(\exists y \leq s)A\). 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 \(\Sigma^1_i\)-replacement axioms are the formulae of the form

\[(\forall x \leq t)(\exists y \leq s)A(x,y) \iff (\exists w \leq S(y,v,t))(\forall x \leq t)A(x,\delta(S(x,z),w))\delta(S(x,z),w) \leq s)\]

where \(s\) and \(t\) are arbitrary terms and \(A\) is any \(\Sigma^1_i\)-formula, and other free variables may appear in \(A\).

**Theorem 14:** Let \(i \geq 1\). Then the \(\Sigma^1_i\)-replacement axioms are theorems of \(S^1_2\).

**Proof:** Let \(A\) be any \(\Sigma^1_i\)-formula. Let \(Y\) and \(Z\) be the formulae

\[Y = (\forall x \leq t)(\exists y \leq s)A(x,y)\]
\[Z(u) = (\exists w \leq S(y,v,t))(\forall x \leq t)(x \leq w \supset A(x,\delta(S(x,z),w))\delta(S(x,z),w) \leq s)\]

We want to show \(S^1_2 \vdash Y \supset Z(t)\). Now, \(S^1_2 \vdash Z(t) \supset Y\) is obvious. Also,

\(S^1_2 \vdash Y \supset Z(0)\)

and

\(S^1_2 \vdash Y \supset Z(u) \iff u < |t| \supset Z(S(u))\).

Thus, by \(\Sigma^1_i\)-LIND, \(S^1_2 \vdash Y \supset Z(t)\).

Q.E.D.

**Definition:** The sets \(\Sigma^1_i(AS)\) and \(\Pi^1_i(AS)\) are defined inductively by:

1. \(\Sigma^1_i(AS)\) is the set of \(\Pi^1_i\)-formulae which are \(\Delta^1_i\) with respect to the theory \(S^1_2\). Similarly, \(\Pi^1_i(AS)\) is the set of \(\Sigma^1_i\)-formulae which are \(\Delta^1_i\) with respect to the theory \(S^1_2\).
(2) $\Sigma^b_{i+1}(\mathit{AS})$ is the smallest set satisfying:
(a) $\Sigma^b_{i+1}(\mathit{AS}) \supseteq \Pi^b_i(\mathit{AS})$ and
(b) If $A \in \Sigma^b_{i+1}(\mathit{AS})$ then $(\exists x \leq t)A$ is in $\Sigma^b_{i+1}(\mathit{AS})$.

(3) $\Pi^b_{i+1}(\mathit{AS})$ is the smallest set satisfying:
(a) $\Pi^b_{i+1}(\mathit{AS}) \supseteq \Sigma^b_i(\mathit{AS})$ and
(b) If $A \in \Pi^b_{i+1}(\mathit{AS})$ then $(\forall x \leq t)A$ is in $\Pi^b_{i+1}(\mathit{AS})$.

The $(\mathit{AS})$ means alternative sense. Note that $\Sigma^b_i$ is a proper subset of $\Sigma^b_i(\mathit{AS})$ and that $\Sigma^b_{i+1}(\mathit{AS})$ is a proper subset of $\Sigma^b_{i+1}$.

Let $R_i$ be the theory $S^L_i$ plus the $\Sigma^b_i$-replacement axioms.

**Corollary 15:** If $A$ is a $\Sigma^b_i$- or a $\Pi^b_i$-formula, then there is a $\Sigma^b_i(\mathit{AS})$- or a $\Pi^b_i(\mathit{AS})$-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 $\beta$ is $\Sigma^b_i$-definable. Theorem 14 asserts that $S^L_i \vdash R_i$. Although we don’t know if the converse is true, we do have the following theorem:

**Theorem 16:** $R_{i+1} \vdash S^L_i$.

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

Let $A$ be any $\Sigma^b_i$-formula. We want to show

$$R_{i+1} \vdash A(0) \land (\forall x)(A(x) \rightarrow A(Sx)) \land (\forall x)(A(\beta x))$$

By Corollary 15, there is a $\Sigma^b_i(\mathit{AS})$-formula $B$ such that $R_i \vdash A(x) \rightarrow B(x)$. Let $B$ have the form

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

where $C \in \Pi^b_i$. 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

$$D(x, y) = C(x, y) \land y \leq t(x).$$

Let $u$ be a new variable. Then, by prenex operations,
§2.7 Replacement Axioms

\[ R_{+1} \vdash (\forall x)(B(x) \supset B(Sx)) \supset (\forall x < \lceil w \rceil) (\exists y \leq t(Sx)) (\forall z \leq t(x)) D(x, z) \supset D(Sx, y). \]

By \(\Sigma^1_{+1}\)-replacement,

\[ R_{+1} \vdash (\forall x)(B(x) \supset B(Sx)) \supset (\exists w)(\forall x < \lceil w \rceil) D(x, z) \supset D(Sx, \beta(Sx, w)). \]

Let \( f \) be the \(\Sigma^1_{+1}\)-definable function satisfying

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

Thus,

\[ R_{+1} \vdash D(0, b) \cdot (\forall x)(B(x) \supset B(Sx)) \supset (\exists w)(\forall y < \lceil w \rceil) D(x, f(w, b), y) \supset D(Sx, \beta(Sx, y)). \]

So,

\[ R_{+1} \vdash B(0) \cdot (\forall x)(B(x) \supset B(Sx)) \supset (\exists w) (D(0, \beta(1, w)) \cdot (\forall y < \lceil w \rceil) D(x, \beta(x+1, w)) \supset D(Sx, \beta(x+2, w))). \]

Since \( D \in \Pi^1_{+1} \), we can use \( \Pi^1_{+1}-\text{LIND} \), to get

\[ R_{+1} \vdash B(0) \cdot (\forall x)(B(x) \supset B(Sx)) \supset (\exists w) D([w], \beta([x]+1, w)). \]

Note that we are justified in using \( \Pi^1_{+1}-\text{LIND} \) by our induction hypothesis and by Theorem 13. Finally,

\[ R_{+1} \vdash B(0) \cdot (\forall x)(B(x) \supset B(Sx)) \supset B([w]). \]

Q.E.D. \( \square \)

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

\[ (\exists x)A(x) \supset (\exists x)[A(x) \land (\forall y \leq x)(y \neq x \supset \lnot A(y))] \]

where A is any formula in Ψ.

The Ψ-LMIN axioms are given by the schema

\[ (\exists x)A(x) \supset A(0) \lor (\exists x)[A(x) \land (\forall y \leq \lfloor \frac{1}{2} x \rfloor)(\lnot A(y))] \]

where again A ∈ Ψ.

Theorem 17: Let \( i \geq 1 \). In the theory \( S^1_i \),
(a) \( \Sigma_i^1 \text{-MIN} \) is equivalent to \( \Pi_i^1 \text{-IND} \), and
(b) \( \Sigma_i^1 \text{-LMIN} \) is equivalent to \( \Pi_i^1 \text{-PIND} \).

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

First, we show that \( \Sigma_i^1 \text{-LMIN} \implies \Pi_i^1 \text{-PIND} \). Let \( A \in \Pi_i^1 \). Then by \( \Sigma_i^1 \text{-LMIN} \),

\[ \lnot (\forall x)A(x) \supset \lnot A(0) \lor (\exists x)(\lnot A(x) \land (\forall y \leq \lfloor \frac{1}{2} x \rfloor)(A(y))) \]

and thus

\[ \lnot (\forall x)A(x) \land A(0) \supset (\exists x)(A(\lfloor \frac{1}{2} x \rfloor) \lor \lnot A(x)) \]

which is what we needed to show.

Secondly, we show \( \Pi_i^1 \text{-PIND} \implies \Sigma_i^1 \text{-LMIN} \). Let \( A(x) \) be a \( \Sigma_i^1 \)-formula. Let \( B(x) \) be the formula \( (\forall y \leq x)(\lnot A(y)) \). Now, by \( \Pi_i^1 \text{-PIND} \),

\[ \lnot B(0) \supset \lnot B(0) \lor (\exists x)(B(\lfloor \frac{1}{2} x \rfloor) \lor \lnot B(x)) \]

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

\[ A(u) \land A(0) \lor (\exists x)(A(\lfloor \frac{1}{2} x \rfloor) \lor (\exists y \leq x)(A(y))). \]

Since the BASIC axioms imply \( y \leq x \supset \lfloor \frac{1}{2} y \rfloor \leq \lfloor \frac{1}{2} x \rfloor \) we get

\[ A(u) \supset A(0) \lor (\exists y \leq u)(\forall x \leq \lfloor \frac{1}{2} x \rfloor)(\lnot A(x) \land A(y)). \]

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 $\Sigma^P_1$-$\text{MIN}$ and $\Pi^P_1$-$\text{MIN}$ are equivalent with respect to a simple open theory $\mathcal{O}$. However in Bounded Arithmetic we have a different situation.

**Theorem 18:** Let $i \geq 1$. $S^i_1+\Pi^i_1$-$\text{MIN}$ is equivalent to $S^i_1+\Sigma^i_{i+1}$-$\text{MIN}$.

**Proof:** Since $\Pi^i_1 \subseteq \Sigma^i_{i+1}$, one direction is trivial. We need to show that $\Pi^i_1$-$\text{MIN} \implies \Sigma^i_{i+1}$-$\text{MIN}$ in the presence of $S^i_1$. We begin by showing that $S^i_1+\Pi^i_1$-$\text{MIN}$ proves the $\Sigma^i_{i+1}$-$\text{MIN}$ axioms.

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

$$(\exists y_1 \leq t_1(x)) \cdots (\exists y_a \leq t_a(x)) B(x,y_1,\ldots,y_a)$$

where $B \in \Pi^i_1$ (since $i \geq 1$). We can assume without loss of generality that the terms $t_i$ do not include the variables $y_j$. Let $B^*(x,y_1,\ldots,y_a)$ be the formula

$$B(x,y_1,\ldots,y_a) \land y_1 \leq t_1(x) \land \cdots \land y_a \leq t_a(x).$$

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

$$P\text{SqSL}(w,a,n+1) \land B^*(\text{Proto}(n+1,w),\text{Proto}(n,w),\ldots,\text{Proto}(1,w)).$$

By Theorem 2, $C$ is $S^i_1$-provably equivalent to a $\Pi^i_1$-formula. $C$ asserts that $w$ is a protosequence coding values for $x$ and $y_1$ which witness that $(\exists x)A(x)$ is true. Now,

$$S^i_1 \vdash (\exists x)A(x) \iff a = \max ([x],[t_1(x)],\ldots,[t_a(x)]) \iff \exists a C(w,a).$$

Since protosequence code entries as fixed length codes,

$$S^i_1 \vdash P\text{SqSL}(w,a,n+1) \land P\text{SqSL}(v,a,n+1) \land w \geq v \implies \text{Proto}(n+1,w) \geq \text{Proto}(n+1,v).$$

So by applying $\Pi^i_1$-$\text{MIN}$, we get a minimum value for $w$ which satisfies $C(w,a)$ (a is held constant). But now $\text{Proto}(n+1,w)$ gives a minimum value for $x$ satisfying $A(x)$. This completes the proof that $S^i_1+\Pi^i_1$-$\text{MIN}$ proves the $\Sigma^i_{i+1}$-$\text{MIN}$ axioms.

To finish the proof of our current theorem, we must show that $S^i_1+\Sigma^i_{i+1}(AS)$-$\text{MIN}$ proves the $\Sigma^i_{i+1}$-$\text{MIN}$ axioms. It will suffice to show that $S^i_1+\Sigma^i_{i+1}(AS)$-$\text{MIN}$ proves the $\Sigma^i_{i+1}$-replacement axioms, since by Corollary 15 a $\Sigma^i_{i+1}$-formula is equivalent to a
Lemma 19: Let \( i \geq 0 \). \( \Sigma^i_1(AS) \)-replacement. The proof of Theorem 17(a) shows that \( \Sigma^i_1(AS) \rightarrow \Pi^i_1(AS) \)-IND. Also, \( S^i_1 + \Pi^i_1(AS) \rightarrow \Sigma^i_1(AS) \)-IND can be shown by using the proof of Theorem 5 (this depends on the fact that the defining equation for subtraction \((-)\) does not contain any sharply bounded quantifiers.) Clearly, \( \Sigma^i_1(AS) \)-IND \( \rightarrow \) \( \Sigma^i_1(AS) \)-LIND. Hence it suffices to prove the following lemma.

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 \( S^i_1 + \Sigma^i_1(AS) \)-LIND proves that every \( \Sigma^i_1 \)-formula is equivalent to a \( \Sigma^i_1(AS) \)-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 \( \Sigma^j \)-formula which is not provably equivalent to a \( \Sigma^j(AS) \)-formula by \( S^i_1 + \Sigma^i_1(AS) \)-LIND.

We shall now show that if \( \forall x \exists y \in [t] \) then \( B \) is provably equivalent to a \( \Sigma^j \)-formula. It suffices to assume that

\[
B = (\forall x \leq \langle t \rangle)(\exists y \leq \langle t \rangle) \phi(x, y)
\]

where \( \phi \in \Pi^i_1(\mathbb{A}) \), as multiple adjacent existential quantifiers in \( B \) can be combined by use of the \( \beta \) 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

\[
(\exists w \leq S_0 B \langle s, t \rangle)(\forall x \leq \langle t \rangle)(x \leq w \supset A(x, \beta(S_\sigma, w)) \cup \beta(S_\sigma, w) \leq w).
\]

We use the proof of Theorem 14 to prove this. The crucial point of the proof of Theorem 14 used \( \Sigma^i_1 \)-LIND on the formula \( Z(u) \). But how can we use \( \text{LIND} \) on \( Z \)? Well, by our choice of \( j \) and since \( \forall x \exists y \in [t] \), \( S^i_1 + \Sigma^i_1(AS) \)-LIND proves that \( Z \) is equivalent to a \( \Sigma^j \)-formula. Hence we are justified in using \( \text{LIND} \) on the formula \( Z \).

This justifies 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 \( \Sigma^i_1 \)-MIN axioms are theorems of \( S^i_1 \).
Minimization Axioms

§2.8

Proof: Let $A(z)$ be any $\Sigma^b_1$-formula. Let $B(a,b,c)$ be

$$(\forall z)(\exists x)(x < a \rightarrow z < b \rightarrow B(z,c) = 0)(\forall y < c)(\exists y < 2^{a+b}).$$

Clearly,

$S^1_2 \vdash A(a) \supset B(a,0,0) \forall a \rightarrow 0.$

So,

$S^1_2 \vdash A(a) \land a \neq 0 \rightarrow (\exists x < c) B(a,0,x)$

We also claim that

$S^1_2 \vdash A(a) \land a \neq 0 \land b < |a| (\exists x \leq b \supset B(a,b,x) \supset (\exists x \leq a) B(a,Sb,x)).$

This is true because

$S^1_2 \vdash b < |a| \rightarrow B(a,b,c) \land (\exists y < 2^{b+c}) A(c+y) \supset B(a,Sb,c)$

and

$S^1_2 \vdash b < |a| \rightarrow (\exists y < 2^{b+c}) A(c+y) \supset B(a,Sb,c) \rightarrow (\exists x \leq a) B(a,Sb,x).$

These last two results follow from the bit manipulation techniques developed while bootstrapping $S^1_2$. Finally, from the definition of $B$ we have

$S^1_2 \vdash A(a) \land B(a,b,c) \supset c \leq a.$

Putting the above results together proves the claim.

Since $B$ is a $\Sigma^b_1$-formula, we can use $\Sigma^b_1$-LIND on the formula $(\exists x \leq a) B(a,b,x)$ to get

$S^1_2 \vdash A(a) \land a \neq 0 \rightarrow (\exists x \leq a) B(a,a,x).$

From this the $\Sigma^b_1$-MIN axiom for $A$ is immediate.

Q.E.D.

Corollary 21: If $i \geq 0$, $S_i^{k+1} \vdash T_i^1$.

Proof: For $i \geq 1$, this follows from Theorems 20, 17(a) and 13(c). For $i = 0$, this is a corollary to the next theorem. Q.E.D.
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 $\Delta^1_i$ formulae are those provably equivalent to both a $\Sigma^1_i$- and a $\Pi^1_i$-formula.

**Theorem EE:** Let $i \geq 1$. The $\Delta^1_i$-IND axioms are theorems of $S^1_i$. ($\Delta^1_i$ means with respect to $S^1_i$.)

**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 $A^L$ in $\Sigma^1_i$ and $A^R$ in $\Pi^1_i$ such that $S^1_i \vdash A^L \leftrightarrow A^R$ and $S^1_i \vdash A^L \leftrightarrow A^R$. Let $B(x,z)$ be the formula

$$(\forall y \leq Sz)(A(x \cdot y) \supset A(x))$$

so $B$ is provably equivalent to a $\Pi^1_i$-formula. We claim that

$$S^1_i \vdash (\forall x \leq c)B(x,[1,\ldots,d]) \supset (\forall x \leq c)B(x,d)$$

where $c$ and $d$ are new free variables. This is because $A(x \cdot y) \supset A(x)$ follows from $A(x \cdot y) \supset A(x \cdot [1,\ldots,y])$ and $A(x \cdot [1,\ldots,y]) \supset A(x)$. So by $\Pi^1_i$-PIND,

$$S^1_i \vdash (\forall x \leq c)B(x,0) \supset (\forall x \leq c)B(x,c).$$

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 $S^1_i$. Hence, $S^1_i$ 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 $i \geq 0$, $T_2^{i+1} \iff S_2^{i+1}$ and $S_2^{i+1} \iff T_2^i$.


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

(a) $\Sigma_{i+1}^1$-IND $\iff$ $\Pi_{i+1}^1$-IND $\iff$ $\Sigma_{i+1}^1$-MIN

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

(c) $\Sigma_{i+1}^1$-replacement $\iff$ $\Sigma_i^1$- replacement.

Proof: By Theorems 5, 6, 11, 13, 14, 16, 17, and 18 and Corollary 21. □
Chapter 8

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 so say, with the theories $S^F_i$.

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 $S^F_i$ 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 $\Sigma^P_i$, $\Pi^P_i$ and $\Delta^P_i$, where $\Sigma^P_i$ is NP and $\Delta^P_i$ 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 $\Omega_i$ formed the polynomial hierarchy of functions. The functions in $\Omega_i$ 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 $\Sigma^P_{i-1}$. For example, $\Omega^P_i$ is the set of functions computable in polynomial time.

**Theorem 5:** Let $k \geq 1$. Let $f$ be an $m$-ary $\Omega^P_i$-function. Let $t(\bar{x})$ be a term (in the language of Bounded Arithmetic) so that for all $x \in \mathbb{N}^m$, $f(\bar{x}) \leq t(\bar{x})$. Then there is a $\Sigma^P^i$-formula $A$ such that

1. $\delta^i_x \vdash \forall \bar{y} \exists \bar{y} \leq t(A(\bar{x}, \bar{y}))$
2. $\delta^i_x \vdash \forall \bar{y} \forall \bar{z} (A(\bar{x}, \bar{y}) \land A(\bar{x}, \bar{z}) \implies \bar{y} = \bar{z})$
3. For all $\forall x \in \mathbb{N}^m$, $A(\bar{x}, f(\bar{x}))$ is true.

Theorem 1 says that the theory $S^P_i$ can $\Sigma^P_{i-1}$-define all of the functions which are polynomial time computable relative to the $\Sigma^P_{i-1}$ 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 $s(\bar{x})$ is another term such that for all $x \in \mathbb{N}^m$, $f(\bar{x}) \leq s(\bar{x})$. Let $B$ be the formula

$$B(\bar{x}, \bar{y}) = (\exists z \leq t)(A(\bar{x}, x) \land y = \min(s, z)).$$
Then,

\[ S_{\mathbb{N}}^1 \vdash (\forall x)(\exists y \leq x)B(x, y) \]
\[ S_{\mathbb{N}}^1 \vdash (\forall x)(\forall y)(\forall w)(B(x, y) \land B(x, w) \supset y = w) \]

and for all \( \exists x \in \mathbb{N}^n, B(x, f(x)) \) is true.

Thus it will suffice to prove that if \( f \in \Sigma^1_1 \) 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 \( \Sigma^1_1 \)-defined by \( S_{\mathbb{N}}^1 \).

**Case (2):** Suppose \( f \) is defined by composition as \( f(x) = g(h_1(x), \ldots, h_n(x)) \) where \( g, h_1, \ldots, h_n \) are functions in \( \Sigma^1_1 \) and that \( S_{\mathbb{N}}^1 \) can \( \Sigma^1_1 \)-define \( g, h_1, \ldots, h_n \) with the formulae

\[ S_{\mathbb{N}}^1 \vdash (\exists z \leq s(g_1, \ldots, g_n))A_1(g_1, \ldots, g_n, z) \]
\[ S_{\mathbb{N}}^1 \vdash (\exists z \leq r(x))A_2(x, z) \]

respectively. Let \( A(x, z) \) be the formula

\[(\exists y_1 \leq r_1(x)) \cdots (\exists y_n \leq r_n(x))(A_1(g_1, \ldots, g_n, z) \land A_2(x, z) \land \cdots \land A_n(x, y_n)).\]

Then \( A \in \Sigma^1_1 \) and for all \( x \in \mathbb{N}, A(x, f(x)) \) is true. Let \( t(x) \) 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. \( PB^2 \)). That is to say,

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

Suppose also that \( g \) is \( \Sigma^1_1 \)-definable by \( S_{\mathbb{N}}^{1-1} \) with the defining condition

\[ S_{\mathbb{N}}^{1-1} \vdash (\forall x)(\forall y)(\exists z \leq \tau(u, x))(A_3(x, y, z)) \]

where \( A_3 \) is a \( \Sigma^1_1 \)-formula. Let \( A(x, z) \) be

\[ [z = 1 \land (\exists u \leq x)(\exists y \leq r)(y \neq 0 \lor A_3(u, x, y))] \lor [z = 0 \land (\forall u \leq x)(\forall y \leq r)(y \neq 0 \supset A_3(u, x, y))]. \]
Then for all values of \( \mathcal{F}, A(\mathcal{F}, f(\mathcal{F})) \) is true. Also \( A \) is clearly \( \Delta^1_1 \). Let \( t(\mathcal{F}) \) 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 \( S^1_2 \)-defined by \( S^1_2 \) by the defining conditions

\[
S^1_2 \vdash (\forall \mathcal{F}) (\exists x \leq s) A(\mathcal{F}, x)
\]

\[
S^1_2 \vdash (\forall \mathcal{F}) (\forall \mathcal{E}) (\exists x \leq r(\mathcal{F}, u, v)) A(\mathcal{F}, u, v, x).
\]

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

\[
\text{Seq}(w) \land \text{Len}(w) = u + 1 \land \beta(1, w) = \min(2^{\xi(\mathcal{F})}, p(\mathcal{F})) \land \beta(i < u) (\beta(i + 2, w) = \min(2^{\xi(\mathcal{F})}, h(\mathcal{F}, i, \beta(i + 1, u))))
\]

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

\[
S^1_2 \vdash (\exists w \leq \text{Seq}(\mathcal{F})) (\exists i \leq s) [B(w, i) \land \beta(i < u) (\beta(i + 2, w) = \min(2^{\xi(\mathcal{F})}, h(\mathcal{F}, i, \beta(i + 1, u)))].
\]

Note that \( B \) is a \( \Sigma^1_1 \)-formula since the quantifier \( (\forall i < u) \) is equivalent to a sharply bounded quantifier. So by \( \Sigma^1_1 \)-PIND,

\[
S^1_2 \vdash (\exists w \leq \text{Seq}(\mathcal{F})) [B(w, s)]
\]

Also, \( S^1_2 \) proves that this sequence \( w \) is unique by the use of \( \Sigma^1_1 \)-LIND on the length of \( w \). So let \( A(\mathcal{F}, y) \) be the formula

\[
(\exists w \leq \text{Seq}(\mathcal{F})) (\beta(1, w) = \min(2^{\xi(\mathcal{F})}, p(\mathcal{F}))) \land g = h(\beta([x] + 1, w)).
\]

Let \( t(\mathcal{F}) \) be \( 2^{\xi(\mathcal{F})} \). Then conditions (1)-(3) hold.

Q.E.D. ☐

We have a similar theorem regarding the definability of \( \Delta^1_k \) predicates in \( S^1_2 \).

**Theorem 2**: Let \( k \geq 1 \). Let \( Q \) be an \( m \)-ary \( \Delta^1_k \) predicate. Then there are formulae \( A \) and \( B \) in \( \Sigma^1_k \) and \( \Pi^1_k \), respectively, so that

1. \( S^1_2 \vdash (\forall \mathcal{F})(A(\mathcal{F}) \rightarrow B(\mathcal{F})) \)
2. For all \( \in \mathbb{N}^m \), \( A(\mathcal{F}) \iff B(\mathcal{F}) \iff Q(\mathcal{F}) \).
Proof: Let $f$ be the $\bigcirc_{f}$ function defined by

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

Let $I(\mathcal{F})$ be the constant term 1. Let $A_f$ be a $\Sigma^k_1$-formula satisfying (1)-(3) of Theorem 1. Define $A$ and $B$ to be

$$A(\mathcal{F}) = A_f(z, 1)$$
$$B(\mathcal{F}) = \neg A_f(z, 0).$$

Then $A \in \Sigma^k_1$ and $B \in \Pi^k_1$ 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 $S^k_1$ with a defined function or predicate symbol and used freely in induction formulae ($i \geq 1$).

**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.


Natural deduction uses the following types of symbols:

1. **Constants**: for example, 0.
2. **Relations**: for example, ≤ and =.
3. **Functions**: for example, S, +, \#, [\#2x], and |*|.
4. **Free variables**: denoted by a, b, c, ...
5. **Bound variables**: denoted by x, y, z, ...
6. **Propositional connectives**: ∧, ∨, →, and ¬.
7. **Bounded quantifiers**: ∀x ≤ and ∃x ≤.
8. **Unbounded quantifiers**: ∀ and ∃.
9. **Parentheses**.
10. **Sequent connective**: →
11. **Comma**.

Terms are built up from constants, free variables and functions. Formulas 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** if it contains no free variables.
§4.1 Syntax and Rules of Natural Deduction

A series of formulae separated by commas is called a cedent. If $\Gamma$ and $\Delta$ are cedents then $\Gamma \rightarrow \Delta$ is a sequent. The antecedent and succedent of $\Gamma \rightarrow \Delta$ are $\Gamma$ and $\Delta$ respectively. The intended meaning of $\Gamma \rightarrow \Delta$ is that the conjunction of the formulae in $\Gamma$ implies the disjunction of the formulae in $\Delta$. Although their meanings are similar, $\supset$ and $\rightarrow$ 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 $x, y, z, \ldots$ 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

\[
\frac{\frac{B}{A}}{ \text{or} } \frac{B \quad C}{A}
\]

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. $\Gamma$, $\Pi$, $\Lambda$ and $\Delta$ are used to denote (parts of) cedents, $A$ and $B$ are arbitrary formulae and $s$ and $t$ are arbitrary terms.

1. (Weak: left)

\[
\frac{\Gamma \rightarrow \Delta}{\Lambda, \Gamma \rightarrow \Delta}
\]

2. (Weak: right)

\[
\frac{\Gamma \rightarrow \Delta}{\Gamma \rightarrow \Delta, A}
\]

3. (Contraction: left)

\[
\frac{A, A \rightarrow \Delta}{A, \Gamma \rightarrow \Delta}
\]

4. (Contraction: right)

\[
\frac{\Gamma \rightarrow \Delta, A, A}{\Gamma \rightarrow \Delta, A}
\]
(5) (Exchange: left)
\[
\frac{\Gamma, A, B, \Pi \rightarrow \Delta}{\Gamma, B, A, \Pi \rightarrow \Delta}
\]

(6) (Exchange: right)
\[
\frac{\Gamma \rightarrow \Delta, A, B, \Pi}{\Gamma \rightarrow \Delta, B, A, \Pi}
\]

(7) (~ left)
\[
\frac{\Gamma \rightarrow \Delta, A}{\neg A, \Gamma \rightarrow \Delta}
\]

(8) (~ right)
\[
\frac{A, \Gamma \rightarrow \Delta}{\Gamma \rightarrow \Delta, \neg A}
\]

(9) (\& left)
\[
\frac{A, \Gamma \rightarrow \Delta}{A \& B, \Gamma \rightarrow \Delta}
\]
and
\[
\frac{A, \Gamma \rightarrow \Delta}{B \& A, \Gamma \rightarrow \Delta}
\]

(10) (\& right)
\[
\frac{\Gamma \rightarrow \Delta, A \quad \Gamma \rightarrow \Delta, B}{\Gamma \rightarrow \Delta, A \& B}
\]

(11) (\lor left)
\[
\frac{A, \Gamma \rightarrow \Delta \quad B, \Gamma \rightarrow \Delta}{A \lor B, \Gamma \rightarrow \Delta}
\]

(12) (\lor right)
\[
\frac{\Gamma \rightarrow \Delta, A}{\Gamma \rightarrow \Delta, A \lor B}
\]
and
\[
\frac{\Gamma \rightarrow \Delta, A}{\Gamma \rightarrow \Delta, B \lor A}
\]
(13) (⊃ left) \[ \frac{\Gamma \rightarrow \Delta, A}{A \supset B, \Gamma, \Pi \rightarrow \Delta, A} \]

(14) (⊃ right) \[ \frac{A, \Gamma \rightarrow \Delta, B}{\Gamma 

\rightarrow \Delta, A \supset B} \]

(15) (∀ left) \[ \frac{A(t), \Gamma \rightarrow \Delta}{(\forall x)A(x), \Gamma \rightarrow \Delta} \]

(16) (∀ right) \[ \frac{\Gamma 

\rightarrow \Delta, A(a)}{\Gamma 

\rightarrow \Delta, (\forall x)A(x)} \]

where \( a \) is a free variable which may not appear in the lower sequent of the inference.

(17) (∃ left) \[ \frac{A(a), \Gamma \rightarrow \Delta}{(\exists x)A(x), \Gamma \rightarrow \Delta} \]

where \( x \) is a free variable which may not appear in the lower sequent of the inference.

(18) (∃ right) \[ \frac{\Gamma 

\rightarrow \Delta, A(t)}{\Gamma 

\rightarrow \Delta, (\exists x)A(x)} \]

(19) (∀ ≤ left) \[ \frac{A(t), \Gamma \rightarrow \Delta}{t \leq s, (\forall x \leq s)A(x), \Gamma \rightarrow \Delta} \]

(20) (∀ ≤ right) \[ \frac{s \leq t, \Gamma \rightarrow \Delta, A(a)}{\Gamma 

\rightarrow \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) \( (\exists \leq \text{left}) \)

\[
\frac{s \leq t, A(a), \Gamma \rightarrow \Delta}{(\exists \leq \text{left}) A(x), \Gamma \rightarrow \Delta}
\]

where \( a \) is a free variable which may not appear in the lower sequent of the inference.

(22) \( (\exists \leq \text{right}) \)

\[
\frac{\Gamma \rightarrow \Delta, A(t)}{s \leq t, \Gamma \rightarrow \Delta, (\exists \leq \text{right}) A(x)}
\]

(23) \( (\text{Cut}) \)

\[
\frac{\Gamma \rightarrow \Delta A}{\Pi \rightarrow \Delta, A \Pi \rightarrow \Delta}
\]

The inferences (1)-(6) are called structural inferences. Rules (7)-(22) are the logical inferences. Rules (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 formulas 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 \( (\exists \leq \leq \text{right}) A(x) \) are the principal formulas of inferences (7) and (22) respectively and their auxiliary formulas are \( A \) and \( A(t) \) respectively.

A logical axiom is a sequent of the form \( A \rightarrow A \) where \( A \) must be an atomic formula. An equality axiom is a sequent of the form \( \rightarrow t_1 = t_2 \).

\[
t_1 = s_1, \ldots, t_n = s_n \rightarrow f(t_1, \ldots, t_n) = f(s_1, \ldots, s_n),
\]

or

\[
t_1 = s_1, \ldots, t_n = s_n, p(t_1, \ldots, t_n) \rightarrow p(s_1, \ldots, s_n)
\]

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 \( \Delta \) 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 \( A \rightarrow A \). So consider the following two LKB proofs:

\[
\begin{align*}
A(a) \rightarrow & \ A(a) \\
\text{\textit{so}} \ A(a) \rightarrow & \ A(a) \\
\text{\textit{so}} \ A(a) \rightarrow & \ A(a) \\
\text{\textit{so}} \ A(a) \rightarrow & \ A(a) \\
\text{\textit{so}} \ A(a) \rightarrow & \ A(a)
\end{align*}
\]

and

\[
\begin{align*}
\text{\textit{so}} \ A(a) \rightarrow & \ A(a) \\
\text{\textit{so}} \ A(a) \rightarrow & \ A(a) \\
\text{\textit{so}} \ A(a) \rightarrow & \ A(a) \\
\text{\textit{so}} \ A(a) \rightarrow & \ A(a)
\end{align*}
\]

\[
\begin{align*}
\text{\textit{so}} \ A(a) \rightarrow & \ A(a) \\
\text{\textit{so}} \ A(a) \rightarrow & \ A(a) \\
\text{\textit{so}} \ A(a) \rightarrow & \ A(a) \\
\text{\textit{so}} \ A(a) \rightarrow & \ A(a)
\end{align*}
\]

So LKB proves \( (\forall x)(x \leq t \rightarrow A(x)) \rightarrow (\forall x)(x \leq t \rightarrow A(x)) \). By similar proofs, LKB proves that \( (\exists x)(x \leq t \rightarrow A(x)) \) is equivalent to \( (\exists x)(x \leq t \rightarrow A(x)) \). 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) $\Sigma_1^1$–IND inference.

\[
\begin{align*}
\Gamma, A(s) & \rightarrow A(Sa), \Delta \\
\Gamma, A(0) & \rightarrow A(t), \Delta
\end{align*}
\]

where $A$ is any $\Sigma_1^1$-formula, $t$ is any term and $a$ is the eigenvariable and must not appear in the lower sequent.

(2) $\Sigma_1^1$–PIND inference.

\[
\begin{align*}
\Gamma, A([a], s^1) & \rightarrow A(s), \Delta \\
\Gamma, A(0) & \rightarrow A(t), \Delta
\end{align*}
\]

with the same proviso as above.

(3) $\Sigma_1^1$–LIND inference.

\[
\begin{align*}
\Gamma, A(s) & \rightarrow A(Sa), \Delta \\
\Gamma, A(0) & \rightarrow A([t], \Delta)
\end{align*}
\]

where, again, the same proviso apply.

If $\Psi$ is any set of formulae, we define the $\Psi$–IND, $\Psi$–PIND and $\Psi$–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 $A$ if $B = A(t_1, \ldots, t_k)$ 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 segments of the form $\rightarrow A$ where $A$ is any substitution instance of an axiom.

Definition:

(a) $S_2^1$ is the natural deduction theory with the BASIC axioms and the $\Sigma_1^1$–PIND induction inferences.

(b) $T_2^1$ is the natural deduction theory with the BASIC axioms and the $\Sigma_1^1$–IND induction inferences.

Theorem $\mathcal{E}$: $(i \geq 0)$. The $\Sigma_1^1$–IND (respectively, $\Sigma_1^1$–PIND, $\Sigma_1^1$–LIND) rule is equivalent to the $\Sigma_1^1$–IND (respectively, $\Sigma_1^1$–PIND, $\Sigma_1^1$–LIND) axioms. Hence the new definitions of $S_2^1$ and $T_2^1$ 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 $\Sigma_1^1$–IND rule can derive the $\Sigma_1^1$–IND axiom and leave the other cases to the reader.
Let $A$ be any $\Sigma_i^1$-formula, and let $a$ and $b$ be any free variables not appearing in $A$. Then we can derive the \textit{IND} axiom for $A$ by:

\[
\begin{align*}
A(a) & \rightarrow A(a) & A(Sa) & \rightarrow A(Sa) \\
A(a) & \rightarrow A(Sa), A(a) & \rightarrow A(Sa) \\
(\forall x)(A(x) \supset A(Sx)), A(a) & \rightarrow A(Sa) \\
(\forall x)(A(x) \supset A(Sx)), A(0) & \rightarrow A(b) \\
(\forall x)(A(x) \supset A(Sx)), A(0) & \rightarrow (\forall x)A(x) \\
A(0)(\forall x)(A(x) \supset A(Sx)), A(0)(\forall x)A(x) & \rightarrow (\forall x)A(x) \\
A(0)(\forall x)(A(x) \supset A(Sx)), A(0) & \rightarrow (\forall x)A(x) \\
\rightarrow A(0)(\forall x)(A(x) \supset A(Sx))(\forall x)A(x)
\end{align*}
\]

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:

\textbf{Definition:} Suppose $C$ is a formula which appears in a given sequent in a proof. The \textbf{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-sequent \( \Gamma, \Delta, \Pi \) or \( \Lambda \) of the upper sequent of an inference, then the successor of \( C \) is the \( k \)-th formula in the corresponding sub-sequent 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 \( D \) if \( C \) 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 sequent 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 2:** (Gentzen) Suppose \( \Gamma \rightarrow \Delta \) is provable in \( S^4 \) or \( T^4 \) by a proof \( P \). Then there is a proof \( P' \) of \( \Gamma \rightarrow \Delta \) 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. \( \Box \)
Corollary 4: (Gentzen) Suppose $\Gamma \rightarrow \Delta$ is provable in LKB. Then $\Gamma \rightarrow \Delta$ 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 $\Gamma$ and $\Delta$ be cedents of $\Sigma_1^i$ and $\Pi_1^i$-formulae and suppose $\Gamma \rightarrow \Delta$ is provable in $T_i^0$ or $T_i^1$. Then there is a proof $P$ of $\Gamma \rightarrow \Delta$ in $S_i^0$ or $T_i^1$ (respectively) such that every formula in $P$ is in $\Sigma_1^i \cup \Pi_1^i$.

Proof: We pick $P$ to be a free cut free proof of $\Gamma \rightarrow \Delta$. Suppose $C$ is a formula in $P$ and that $C \in \Sigma_1^i \cup \Pi_1^i$. Then $C$ cannot 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 $\Sigma_1^i \cup \Pi_1^i$ and this contradicts the hypotheses of the theorem. Thus all formulae in $P$ must be in $\Sigma_1^i \cup \Pi_1^i$. $\square$

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 $\Gamma \rightarrow \Delta$. The free variables in $\Gamma \rightarrow \Delta$ 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 infer-
ence 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 if
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),
($\forall$ left), ($\exists$ right), ($\exists$ 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), ($\forall$ 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 ($\forall$ 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 if $P$ 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 ($\forall$ right) inference.

**Proposition 8:**
(a) Suppose $P$ is a proof of $\Gamma \rightarrow \Delta$. Then there is a proof $P^*$ of $\Gamma \rightarrow \Delta$ such that $P^*$ is in
free variable normal form

(b) Suppose $P$ is a proof of $\Gamma \rightarrow \Delta$. Then there is a proof $P^*$ of $\Gamma \rightarrow \Delta$ 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 $\Gamma \rightarrow \Delta$. 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

$$
\frac{\Gamma, A([t_1/a_1]) \rightarrow A(t_1), \Delta}{\Gamma, A(0) \rightarrow A(t_1), \Delta}
$$

or

$$
\frac{\Gamma, A(a) \rightarrow A(Sa), \Delta}{\Gamma, A(0) \rightarrow A(t_1), \Delta}
$$

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 $\Gamma \rightarrow \Delta$ be a bounded sequent which is provable in one of the theories $S_2$ or $T_2$. Then there is a bounded proof of $\Gamma \rightarrow \Delta$ in the same theory which has no free $\sigma$'s, is in free variable normal form and is restricted by parameter variables.

Before proving Theorem 9, we introduce a new metafunction $\sigma$ which lets the proof apply to slightly more general theories. As a bonus, the use of $\sigma$ may make the proof somewhat easier to understand.

Let $\mathcal{R}$ be any theory of arithmetic. We define a metafunction $\sigma$ which maps terms of the language of $\mathcal{R}$ to terms. Suppose $t_1, \ldots, t_k$ are terms with variables $a_1, \ldots, a_k$. Then $\sigma(t_1, \ldots, t_k)$ is a term with the same variables. Furthermore, if $1 \leq i \leq k$,

$$
b_1 \leq a_{i_1}, \ldots, b_n \leq a_{i_n} \rightarrow t(b_1, \ldots, b_n) \leq \sigma(t_1, \ldots, t_k)(a_{i_1}, \ldots, a_{i_n})
$$

must be provable from the axioms of $\mathcal{R}$ without the use of any induction inferences.

Obviously the metafunction $\sigma$ depends on the theory $\mathcal{R}$, and indeed, for a given theory $\mathcal{R}$ there are many $\sigma$'s satisfying the above conditions. The exact choice for $\sigma$ is not too important, but $\sigma$ should be as simple and as constructive as possible.

If $\mathcal{R}$ is one of the theories $S_2$ or $T_2$, we have a particularly simple definition for $\sigma$.

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 fractions, we can still define \( \sigma \). 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 \( \sigma \).

Theorem 9 is stated only for \( S_2 \) and \( T_2 \), however our use of the \( \sigma \) 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 \( \Gamma \rightarrow \Delta \) 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 \( \epsilon_1, \ldots, \epsilon_p \). 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 \( \epsilon_i \). 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 \( (\forall \leq \text{right}) \) or \( (\exists \leq \text{left}) \). That is, \( J_i \) is either
   
   \[
   \frac{b_i \leq \epsilon_i(b_1, \ldots, b_{i-1}) \Gamma_i, \Delta_i \Rightarrow A(b_i), \Delta_i}{\Gamma_i \Rightarrow (\forall x \leq \epsilon_i(b_1, \ldots, b_{i-1}))A(x), \Delta_i}
   \]

   or
   
   \[
   \frac{b_i \leq \epsilon_i(b_1, \ldots, b_{i-1}), A(b_i), \Gamma_i \Rightarrow \Delta_i}{(\exists x \leq \epsilon_i(b_1, \ldots, b_{i-1}))A(x), \Gamma_i \Rightarrow \Delta_i}
   \]

   where the term \( \epsilon_i \) may contain the free variables \( b_1, \ldots, b_{i-1} \) and may also contain the parameter variables \( \epsilon_1, \ldots, \epsilon_{i-1} \). 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

$$
\frac{\Gamma \vdash A_i[b_i] \rightarrow A_i[b_i], \Delta_i}{\Gamma \vdash A_i(0) \rightarrow A_i[b_i], \ldots, b_{i+1}], \Delta_i}
$$

Again, define $u_i = \sigma_i[x_i][u_{i_1}, \ldots, u_{i_m}]$.

$P$ 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 $\Pi \rightarrow \Lambda$ be a sequent in $P$. Let $b_{i_1}, \ldots, b_{i_m}$ be the free variables of $P$ which have elimination inference below $\Pi \rightarrow \Lambda$. Let $\Xi$ be the cedent

$$b_{i_1} \leq u_{i_1}, \ldots, b_{i_m} \leq u_{i_m}$$

The sequent in $P'$ corresponding to $\Pi \rightarrow \Lambda$ is $\Xi, \Pi \rightarrow \Lambda$. So $P'$ is formed from $P$ by adding $b_{i_1} \leq u_{i_1}$ to every sequent above the elimination inference of $b_{i_1}$ 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 these 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

$$b_{i_1} \leq u_{i_1}, \ldots, b_{i_m} \leq u_{i_m}, \Pi \rightarrow \Lambda$$

where $\Pi \rightarrow \Lambda$ is a valid initial sequent. In $P^*$, this initial sequent is replaced by the initial sequent $\Pi \rightarrow \Lambda$ 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

$$\frac{\Xi, \Gamma \rightarrow \Delta, \Lambda}{\Xi, \Gamma, \Xi, \Pi \rightarrow \Lambda}$$

Unless $\Xi$ is the empty cedent, this is not a valid inference. In $P^*$ this inference is replaced by $I^*$.
\[ \begin{align*}
\therefore \Delta & \rightarrow \Delta, A \\
\therefore \Delta & \rightarrow \Delta, A \Pi \\
\therefore \Delta & \rightarrow \Delta, A
\end{align*} \]

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 \((V \leq \text{right})\) elimination inference \( J_i \). The corresponding inference \( J'_i \) in \( P' \) is

\[ \begin{align*}
b_i \leq u_i, \Xi_{b_i} & \leq s_i \Gamma_i \rightarrow A(b_i), \Delta_i \\
\Xi_{b_i} & \rightarrow (\forall x \leq s_i) A(x), \Delta_i
\end{align*} \]

where \( \Xi_i \) is the cedent containing the formulas \( b_i \leq u_i \) for all \( b_i \) with elimination inference below \( J_i \) in \( P \). Clearly, \( J'_i \) is not a valid inference. In \( P' \) we replace \( J_i \) by \( J'_i \): \( \Xi_{b_i} \leq s_i \rightarrow k_i \leq u_i \\
\therefore \Xi_{b_i} & \leq s_i \Xi_{b_i} \leq s_i \Gamma_i \rightarrow A(b_i), \Delta_i \\
\Xi_{b_i} & \rightarrow (\forall x \leq s_i) A(x), \Delta_i
\end{align*} \]

The first inference is a Cut inference. The sequent \( \Xi_{b_i} \leq s_i \rightarrow b_i \leq u_i \) is provable by the definition of the \( \sigma \) 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 \((\exists \leq \text{left})\) 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:

\[ \begin{align*}
b_i \leq u_i, \Xi_{b_i} & \rightarrow A[b_i], \Delta_i \\
\Xi_{b_i} & \rightarrow A(0), \Delta_i
\end{align*} \]

Clearly this is not a valid inference and in \( P' \) we replace it by \( J'_i \).
where \( d_i \) is a new free variable and the sequents (\( \beta \)), (\( \gamma \)), and (\( \delta \)) used in the Cut inferences are:

\[
\begin{align*}
(\beta) & \quad b_i \leq d_i \rightarrow \forall x \leq d_i, A(x), \\
(\gamma) & \quad A(0) \rightarrow (\forall x \leq 0, y \leq A(x)) \\
(\delta) & \quad \exists y, (x < u_i \rightarrow A(x)) \rightarrow A(s_i)
\end{align*}
\]

Note that these cuts are free since the cut formula is a direct descendant 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. \( \Box \)

It is not at all obvious that Theorem 9 holds for the theories \( S^*_i \) and \( T^*_i \) instead of \( S_i \) and \( T_i \). In fact, it almost certainly does not hold for \( S^*_i \) and \( T^*_i \). However, it does hold for \( S^*_1 \) and \( T^*_1 \) when \( r > 1 \). The author surmises (without proof) that it holds for \( S^*_1 \) and \( T^*_1 \), 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.

This section establishes the following result: Suppose $A(\mathfrak{I})$ is a bounded formula provable in $S_2$ where $\mathfrak{I}$ indicates all the free variables of $A$. Then there is a deterministic polynomial time algorithm $P$ such that for all $\mathfrak{I} \in \mathbb{N}^p$, $P(\mathfrak{I})$ is the Gödel number of a proof of $A(I_{n_1}, \ldots, I_{n_p})$, where the proof $P(\mathfrak{I})$ is bounded and contains no induction inferences. To restate this informally, we can say that if $A$ is bounded and if $S_2 \vdash (\forall \mathfrak{I})A(\mathfrak{I})$ then for each $n$, there is a "short," bounded, induction free proof of $A(\mathfrak{I})$.

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 $S^2_2$.

**Theorem 10:** Let $\Gamma \rightarrow \Delta$ be a bounded sequent provable in $S_2$. Let $a_1, \ldots, a_p$ be the free variables in $\Gamma \rightarrow \Delta$. Then there is a $p$-ary polynomial time function $f$ such that for all $\mathfrak{n} \in \mathbb{N}^p$, $f(\mathfrak{n})$ is the Gödel number of an $S_2$ proof of $\Gamma(a_1, \ldots, a_p) \rightarrow A(a_1, \ldots, a_p)$ which is bounded, does not contain any induction inferences and is in free variable normal form.

Recall that $I_\mathfrak{n}$ is a term with value $\mathfrak{n}$ such that the length of $I_\mathfrak{n}$ is proportional to $|\mathfrak{n}|$. The theorem would certainly be false if $S^0_1 \mathfrak{n}$ were used instead of $I_\mathfrak{n}$ since the length of $S^0_1 \mathfrak{n}$ is exponential in the length of $\mathfrak{n}$.

**Proof:** By Theorem 9 there is a bounded proof $P$ of $\Gamma \rightarrow \Delta$ 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

$$\Gamma, A(I_\mathfrak{m}) \rightarrow A(I_\mathfrak{c}) \Delta$$

where the only free variables in $\mathfrak{c}$ are the $\mathfrak{I}$. We eliminate the induction inference by replacing it with $2(|\mathfrak{m}| + 1)$ Cut inferences. Specifically, if $\mathfrak{m}$ is the value of $t(\mathfrak{m})$, form the $|\mathfrak{m}| + 1$ terms $I_{t(\mathfrak{m})}, I_{t(MP(\mathfrak{m}))}, \ldots, I_{t(n_p)}$. By the induction hypothesis, there is a deterministic polynomial time function $f(\mathfrak{m}, \mathfrak{I})$ which computes the Gödel number for an induction free, bounded proof in free variable normal form of $\Gamma, A(I_\mathfrak{m}) \rightarrow A(I_\mathfrak{c}) \Delta$. By invoking $\mathfrak{h}$ repeatedly we can obtain proofs of each of the sequents

$$\Gamma, A(I_\mathfrak{m}) \rightarrow A(I_{MP(\mathfrak{m}, \mathfrak{I})}) \Delta.$$

It is also easy to construct a proof of $A(I_{MP(\mathfrak{m}, \mathfrak{I}+1)}) \rightarrow A(I_{MP(\mathfrak{m}, \mathfrak{I})})$ for all $\mathfrak{I}$. Then we join
these sequents together with 2·m+i cuts (and a list of exchanges and contractions) to obtain a proof of $\Gamma, A(0) \rightarrow A(I_m)$. Since for every term $t$ there is a polynomial $p_t$ such that $p_t(I) \geq |t|$, 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 $g_t(I)$ is the Gödel number of an induction free proof of $I_m = t(I_a, \ldots, I_n)$. We shall prove this last sentence as part of Lemma 7.3. Thus there is a polynomial time procedure which produces an induction free proof of

$$A(I_m) \rightarrow A(t(I_a, \ldots, I_n)).$$

We combine this with the proof of $\Gamma, A(0) \rightarrow A(I_m).$ obtained above. This yields an induction free, bounded proof of $\Gamma, A(0) \rightarrow A(t(I_a, \ldots, I_n)).$ By renaming 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 $t \geq 0$. Suppose that $A$ is a bounded formula and that $S^i_t$ or $T^i_t$ proves $(\exists y)(\forall x)A(x, y)$. Then there is a term $\tau(x)$ such that the same theory proves $(\forall x)(\exists y \leq \tau(x))A(x, y)$.

**Proof:** By Proposition 8 there must be a free cut free proof $P$ in free variable normal form of the sequent $\rightarrow (\exists y)A(x, y)$. It is easily seen that every formula in $P$ is either $(\exists y)A(x, y)$ or is bounded. Furthermore, every occurrence of $(\exists y)A(x, y)$ is in the antecedent. Thus the only inferences in $P$ involving unbounded quantifiers are (Erasure) inferences which introduce the formula $(\exists y)A(x, y)$.

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 (Erasure) 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*). $P^c$ is defined from $P$ by Cases (1*)-(3*). Cases (1*)-(4*) are the same as (1*)-(4*). The fifth case is:

(5*) Suppose the inference $J_i^c$ in $P^c$ is:

$$
\frac{b_i \leq u_i \in u_i \in \Gamma_i A_i(\lfloor k \rfloor) \rightarrow A_i(b_i) \Delta_i}{\Xi_i \Gamma_i A_i(\emptyset) \rightarrow A_i(s_i) \Delta_i}
$$

Clearly this is not a valid inference and in $P^{c'}$ we replace it by $J_i^c$:

$$
\frac{b_i \leq u_i \in u_i \in \Gamma_i A_i(\lfloor k \rfloor) \rightarrow A_i(b_i) \Delta_i}{\Xi_i \Gamma_i A_i(0) \rightarrow A_i(s_i) \Delta_i}
$$

where (γ) is the sequent $A_i(0) \rightarrow [\lfloor 0 \rfloor \leq u_i \in \Delta_i(\lfloor 0 \rfloor)]$ and (δ) represents

$$
\Xi_i \Delta_i A_i(\emptyset) \rightarrow A_i(s_i) \Delta_i
$$

It is easy to verify that $P^{c'}$ is free cut free and in free variable normal form and that the endsequent of $P^{c'}$ is the same as the endsequent of $P$.

Step (δ): Wherever a (∃right) inference occurs in $P^{c'}$, of the form

$$
\Xi_i \Gamma_i \rightarrow \Delta_i A_i(\exists x f(x))
$$

We replace this inference with:

$$
\Xi_i \Gamma_i \rightarrow \Delta_i A_i(\exists x f(x))
$$

$$
\Xi_i \Gamma_i \rightarrow \Delta_i A_i(\exists x f(x))
$$

$$
\Xi_i \Gamma_i \rightarrow \Delta_i A_i(\exists x f(x))
$$

$$
\Xi_i \Gamma_i \rightarrow \Delta_i A_i(\exists x f(x))
$$
We also replace all the descendants of \((\exists y \leq r)[(\exists y \leq r)]A(\bar{v}, y)\) in \(P'\) by \((\exists y \leq r)[(\exists y \leq r)]A(\bar{v}, y)\) as far down as possible: which means all the descendants either down to the end of \(P'\) or down to a contraction inference with \((\exists y)A(\bar{v}, y)\) as principal formula.

**Step (9):** After doing Step (2) as often as possible, we handle contractions. Suppose the modified proof contains

\[
\Xi \Gamma \rightarrow \Delta(\exists y \leq t)A(\bar{v}, y), (\exists y \leq s)A(\bar{v}, y)
\]

\[
\Xi \Gamma \rightarrow \Delta(\exists y)A(\bar{v}, y)
\]

We replace this by first deriving

\[(\exists y \leq s)A(\bar{v}, y) \rightarrow (\exists y \leq s)[s,t]A(\bar{v}, y)\]

and

\[(\exists y \leq t)A(\bar{v}, y) \rightarrow (\exists y \leq s)[s,t]A(\bar{v}, y)\]

We now use two cuts and a contraction to get:

\[
\Xi \Gamma \rightarrow \Delta(\exists y \leq t)A(\bar{v}, y), (\exists y \leq s)A(\bar{v}, y)
\]

\[
\Xi \Gamma \rightarrow \Delta(\exists y \leq s)[s,t]A(\bar{v}, y), (\exists y \leq s)A(\bar{v}, y)
\]

\[
\Xi \Gamma \rightarrow \Delta(\exists y \leq s)[s,t]A(\bar{v}, y), (\exists y \leq r)A(\bar{v}, y)
\]

\[
\Xi \Gamma \rightarrow \Delta(\exists y \leq r)A(\bar{v}, y)
\]

We now replace the descendants of the original formula \((\exists y)A(\bar{v}, y)\) 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 \((\exists y \leq r)A(\bar{v}, y)\) 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

\[A(x,y) = (\forall z)(z=x\rightarrow y=x)\]

Then \(LKB\) proves \((\forall z)(\exists y)A(x,y)\). But there is no term \(r\) of Bounded Arithmetic such that \((\forall z)(\exists y \leq r)A(x,y)\) 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 $\Omega^2_{\text{PTC}}(\Sigma_i^2)$ can be $\Sigma_i^2$-defined in $S_i^2$. Theorem 3.1 was proved by a straightforward construction of the $\Sigma_i^2$-formula from the definition of a $\Omega_i^2$-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 $S_i^2 \vdash (\forall \bar{x})(\exists s)A(\bar{x},s)$ where $A(\bar{x},s)$ is a $\Sigma_i^2$-formula and $\bar{x}$ and $s$ are the only free variables of $A$. Then there is a term $t(\bar{x})$, a $\Sigma_i^2$-formula $B$ and a function $g$ in $\Omega_i^2$ so that

1. $S_i^2 \vdash (\forall \bar{x})(\forall y)(B(\bar{x},y) \supset A(\bar{x},y))$
2. $S_i^2 \vdash (\forall \bar{x})(\forall y)(B(\bar{x},y) \supset y = z)$
3. $S_i^2 \vdash (\forall \bar{x})(\exists s)(B(\bar{x},s))$
4. For all $\bar{t}$, $N \models B(\bar{t}, g(\bar{t})).$

Corollary 2: Suppose $f$ is a function $\Sigma_i^2$-definable by $S_i^2$. Then $f$ is a $\Omega_i^2$-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 $\Sigma_i^2$-formula and $\bar{z}$ is a vector of free variables which includes all the variables free in $A$. We define below a formula $\text{Witness}_{A}(\bar{w}, \bar{z})$ which is $\Delta_i^2$ with respect to $S_i^2$. The definition is by induction on the complexity of $A$.

1. If $A$ is a $\Sigma_i^1$- or a $\Pi_i^1$-formula, then we define

\[ \text{Witness}_{A}(\bar{w}, \bar{z}) \iff A(\bar{z}) \]
(2) If $A$ is $B \land C$, then we define
\[
\text{Witness}^\mathcal{F}_A(w, \bar{x}) \iff \text{Witness}^\mathcal{F}_B(\beta(1, w), \bar{x}) \land \text{Witness}^\mathcal{F}_C(\beta(2, w), \bar{x})
\]

(3) If $A$ is $B \lor C$, then we define
\[
\text{Witness}^\mathcal{F}_A(w, \bar{x}) \iff \text{Witness}^\mathcal{F}_B(\beta(1, w), \bar{x}) \lor \text{Witness}^\mathcal{F}_C(\beta(2, w), \bar{x})
\]

(4) If $A$ is not in $\Sigma^+_1 \cup \Pi^+_1$ and $A[\vec{d}]$ is $(\forall x \leq |\vec{d}|)B(\vec{x}, x)$, then we define
\[
\text{Witness}^\mathcal{F}_A(w, \bar{x}) \iff \text{Seq}(w) : \text{Len}(w) = \nu(\vec{y}) + 1 \land \forall (\forall x \leq |\vec{y}|) \text{Witness}^\mathcal{F}_B(\beta(x+1, w), \bar{x}, \bar{y})
\]
Thus $w$ witnesses $A[\vec{x}]$ iff $w = \langle w_0, \ldots, w_m \rangle$ and each $w_i$ witnesses $B(\vec{x}, x_i)$.

(5) If $A$ is not in $\Sigma^+_1 \cup \Pi^+_1$ and $A$ is $(\exists x \leq |\vec{d}|)B(\vec{x}, x)$, then we define
\[
\text{Witness}^\mathcal{F}_A(w, \bar{x}) \iff \text{Seq}(w) \land \text{Len}(w) - 2A(1, w) \leq |\vec{x}| \land \forall (\exists x \leq |\vec{d}|) \text{Witness}^\mathcal{F}_B(\beta(2, w), \bar{x}, \beta(1, w))
\]
So $w$ witnesses $A$ if $w = \langle n, \vec{e} \rangle$ where $n \leq t$ and $\vec{e}$ witnesses $B(\vec{x}, n)$.

(6) If $A$ is not in $\Sigma^+_1 \cup \Pi^+_1$ and $A$ is $\neg B$, then we define $\text{Witness}^\mathcal{F}_A$ by using logical prefix operations to transform $A$ so that it can be handled by cases (1)-(5). Specifically, if $A = \neg A'$, then $A \leftrightarrow \neg A'$, $(\forall x \leq t) B$ or $(\exists x \leq t) B$ then $A'$ be $B$, $(\neg A) \leftrightarrow (\neg A')$, $(\neg A) \land (\neg A')$, $(\exists x \leq t) \neg B$ or $(\forall x \leq t) \neg B$ respectively. Then
\[
\text{Witness}^\mathcal{F}_A(w, \bar{x}) \iff \text{Witness}^\mathcal{F}_{A'}(w, \bar{x})
\]
The idea behind defining $\text{Witness}^\mathcal{F}_A$ is that having a $w$ such that $\text{Witness}^\mathcal{F}_A(w, \bar{x})$ is a canonical way of verifying that $A[\vec{x}]$ is true. It is not difficult to see that $(\forall w) \text{Witness}^\mathcal{F}_A(w, \bar{x})$ is equivalent to $A[\vec{x}]$ when $A \in \Sigma^+_1$. Indeed, this is provable by $S^1_2$.

**Proposition 3**: Let $i \geq 1$. Let $A$ be a $\Sigma^+_i$-formula with free variables among $\bar{x}$. Then:

(a) $S^i_1 \vdash \text{Witness}^\mathcal{F}_A(w, \bar{x}) \supset A[\vec{x}]$

(b) There is a term $t_A$ such that
Theorem 4.1: Let \( \mathcal{A} = \langle W, \leq, \langle \mathcal{G}, \mathcal{H} \rangle \rangle \) be a model of \( \mathcal{L}_k \). Then \( \mathcal{A} \vdash \mathcal{L}_k \langle \mathcal{G}, \mathcal{H} \rangle \).

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 (8) of the definition of \( \mathcal{L}_k \) are easily handled. The other two cases are as follows:

Case (5): \( A = \mathcal{L}_k \mathcal{A}^1 \cup \mathcal{L}_k \mathcal{I}^1 \) and \( A \) is \( \exists \mathcal{A} \leq \mathcal{I} \). We argue informally in \( \mathcal{L}_k \). Suppose \( B(\mathcal{A}, \mathcal{I}) \) holds with \( x \leq t \). By the induction hypothesis, we know that there exists a \( \mathcal{A} \leq \mathcal{I} \), such that \( \mathcal{A} \leq \mathcal{I} \). So let \( w \leq x, v \). Then \( \mathcal{L}_k \mathcal{I} \) holds. We can define

\[ t_A(\mathcal{A}) = \mathcal{L}_k \mathcal{B}(\max(t_B(\mathcal{A}, t(\mathcal{I})), t(\mathcal{I}))), t(\mathcal{I})) \]

and we are guaranteed that \( w \leq t_A(\mathcal{A}) \).

Case (6): \( A = \mathcal{L}_k \mathcal{A}^1 \cup \mathcal{L}_k \mathcal{I}^1 \) and \( A \) is \( \forall \mathcal{A} \leq \mathcal{I} \). The induction hypothesis is that

\[ \mathcal{L}_k \mathcal{B}(\mathcal{A}, \mathcal{I}) \leq \mathcal{L}_k \mathcal{A} \leq \mathcal{I} \]

Since the \( \mathcal{L}_k \mathcal{I} \)-replacement axioms are theorems of \( \mathcal{L}_k \) (by Theorem 2.14), it follows that

\[ \mathcal{L}_k \mathcal{A} \leq \mathcal{I} \leq \mathcal{L}_k \mathcal{B}(\max(t_B(\mathcal{A}, t(\mathcal{I})), t(\mathcal{I}))), t(\mathcal{I})) \]

(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. \( \Box \)

Another crucial property of \( \mathcal{L}_k \) is that it is relatively easy to tell whether \( \mathcal{L}_k \mathcal{I} \) holds for arbitrary \( w \) and \( \mathcal{I} \). This is formalized by the next proposition.

Proposition 4: Let \( \mathcal{I} \geq 1 \) and \( A(\mathcal{I}) \leq \mathcal{I}^1 \). Let \( p \) be the predicate defined by

\[ p(w, \mathcal{I}) \iff \mathcal{L}_k \mathcal{I}(w, \mathcal{I}) \]
Then $p$ is a $\Delta^f_1$-predicate (of the polynomial hierarchy).

**Proof:** This is easily proved by induction on the complexity of $A$. 

In particular, when $i = 1$ $p(w, \mathcal{F})$ is a polynomial time predicate. This should not be surprising since if $A$ is a fixed $\Sigma^1_i$-formula it is certainly reasonable that a polynomial time algorithm can check whether $w$ and $\mathcal{F}$ code an instantiation for $A$ which satisfies $A$. Of course, this polynomial time algorithm depends on $A$.

If $\Gamma$ is a cedent we write $\mathcal{A} \Gamma$ (respectively, $\mathcal{V} \Gamma$) to denote the conjunction (respectively, disjunction) of the formulae in $\Gamma$. We adopt the convention that conjunction and disjunction associate from right to left. Thus, if $\Gamma = A, B, C$ then $\mathcal{A} \Gamma$ means $A \land (B \lor C)$. We use the notation

$$<a_1, \ldots, a_n>$$

to denote $<a_1, <a_2, \ldots, <a_{n-1}, a_n> \ldots >$.

These conventions allow us to conveniently discuss witnessing a cedent. For example, suppose $\Gamma$ is $A_1, \ldots, A_n$ and that $w = <w_1, \ldots, w_n>$. Then $\text{Witness}^{\mathcal{V} \Gamma}(w,\mathcal{E})$ holds iff $\text{Witness}^{\mathcal{V} \Gamma}(w_j,\mathcal{E})$ 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 $S^i_f : \Gamma, \Pi \rightarrow A, \Delta$ and that each formula in $\Gamma \cup \Delta$ is a $\Sigma^1_i$-formula and each formula in $\Pi \cup A$ is a $\Pi^1_i$-formula. Let $c_1, \ldots, c_p$ be the free variables in $\Gamma, \Pi \rightarrow A, \Delta$. Let $G$ and $H$ be the $\Sigma^1_i$-formul
e

$$G = (\mathcal{A} \Gamma) \forall \mathcal{V} (\neg c : C \exists \mathcal{A})$$

and

$$H = (\mathcal{V} \Delta) \exists \mathcal{V} (\neg c : C \exists \Pi).$$

Then there is a function $f$ which is $\Sigma^1_i$-definable in $S^i_f$ such that

1. $f$ is a $\Pi^f_1$-function, and
2. $S^i_f : \text{Witness}^{\mathcal{V} \Gamma}(w,\mathcal{E}) \rightarrow \text{Witness}^{\mathcal{V} \Gamma}(f(w,\mathcal{E}),\mathcal{E}).$
Proof: of Theorem 1 from 5:

The hypothesis of Theorem 1 is that $S^2_1\vdash \exists y \forall z \ A(z,y)$. Hence, by Theorem 4.11 there is a term $f(x)$ such that $S^2_1 \vdash \exists y \leq A(f(x),y)$. We now apply Theorem 5 by letting $\Delta$ be $\exists y \leq A(f(x),y)$ and letting $\Gamma = \Pi = \emptyset$. Theorem 5 asserts that there is an $f$ satisfying (1) and (2). Furthermore this $f$ is $\Sigma^b_1$-definable in $S^2_1$ by $f(x) = d$ if and only if $A(f(x),d)$ for some $A \in \Sigma^b_1$ such that

$$ S^2_1 \vdash (\forall y \exists z \ A(f(x),y)). $$

We need to find $B$ and $g$ satisfying (1)-(4) of Theorem 1. We define

$$ g(x) = \beta(1,f(x)) $$

and

$$ B(x,y) \iff y = \beta(1,f(x)). $$

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 $\Omega^b_1$-function since $f$ is.

Q.E.D. □

Proof: of Theorem 5.

By Proposition 4.8, there is an $S^2_1$-proof $P$ of $\Gamma, \Pi \rightarrow \Delta, \Delta$ 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 $\Sigma^b_1\Pi^b_1$, so is every formula appearing anywhere in $P$. Since all induction inferences in $P$ are $\Sigma^b_1$-$\Pi^b_1$ inferences, the principal formula of each cut inference in $P$ is a $\Sigma^b_1$-formula.

To simplify notation and terminology we shall henceforth assume that $\Pi$ and $\Delta$ are the empty cedent. We can always fulfill this requirement by using ($\neg$ left) and ($\rightarrow$ 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, ($\exists$ left) is dual to ($\forall$ right) and ($\neg$ right) is dual to ($\neg$ left).

The proof of Theorem 5 is by induction on the number of inferences in the proof $P$ of $\Gamma \rightarrow \Delta$ 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 $\Gamma \rightarrow \Delta$ must be either a $BASIC$ axiom, a logical axiom or an equality axiom. In either case every formula in $\Gamma \rightarrow \Delta$ is open. The definition of Witness was that

$$ Witness^T_A \iff A(x) $$

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 \((\sim \text{left})\) or \((\sim \text{right})\). These are "cosmetic" inferences: see the discussion above about assuming that \( \Pi \) and \( \Lambda \) are empty.

**Case (2):** \((\lor \text{left})\). Suppose the last inference of \( P \) is

\[
\frac{B, I^* \rightarrow \Delta}{B \lor C, I^* \rightarrow \Delta}
\]

Let \( D \) be the formula \( B \lor (\overline{A}^*) \) and let \( E \) be \((B \lor C) \lor (\overline{A}^*)\). The induction hypothesis is that there is a \( \Omega^\delta \)-function \( g \) such that \( g \) is \( \Sigma^\delta_1 \)-definable by \( S^I_2 \) and

\[
S^I_2 \vdash \text{Witness}^{B^I}(w, \overline{\epsilon}) \supset \text{Witness}^{\Lambda^i}(g(w, \overline{\epsilon}), \overline{\epsilon}).
\]

Let \( h \) be the function defined by

\[
h(w) = <\beta(1, \delta(1, w)), \beta(2, w) >.
\]

Then \( h \in \Omega^\delta_1 \) and

\[
S^I_2 \vdash \text{Witness}^{B^I}(w, \overline{\epsilon}) \supset \text{Witness}^{B^I}(h(w), \overline{\epsilon})
\]

follows immediately from the definition of \( \text{Witness} \). So define \( f(w, \overline{\epsilon}) = g(h(w), \overline{\epsilon}) \). Then \( f \in \Omega^\delta_1 \), \( f \) is \( \Sigma^\delta_1 \)-definable and

\[
S^I_2 \vdash \text{Witness}^{B^I}(w, \overline{\epsilon}) \supset \text{Witness}^{\Lambda^i}(f(w, \overline{\epsilon}), \overline{\epsilon})
\]

which is what we needed to show.

**Case (8):** \((\lor \text{left})\). Suppose the last inference of \( P \) is

\[
\frac{B, I^* \rightarrow \Delta \quad C, I^* \rightarrow \Delta}{B \lor C, I^* \rightarrow \Delta}
\]

Let \( D \) be the formula \( B \lor (\overline{A}^*) \), let \( E \) be \( C \lor (\overline{A}^*) \) and let \( F \) be \((B \lor C) \lor (\overline{A}^*)\). By the induction hypothesis, there are \( \Sigma^\delta_1 \)-definable functions \( g \) and \( h \) in \( \Omega^\delta_1 \) such that
Computational Complexity of Definable Functions

\[ S_{2^1}^1 \vdash \text{Witness}_{\Sigma^P_1}(g(w, x), x) \]
and
\[ S_{2^1}^1 \vdash \text{Witness}_{\Sigma^P_1}(h(w, x), x). \]

We define \( f(w, x) = \begin{cases} g(<\beta(1, \beta(1, w)), \beta(2, w)>, x) & \text{if } \text{Witness}_{\Sigma^P_1}(\beta(1, \beta(1, w)), x) \\ h(<\beta(2, \beta(1, w)), \beta(2, w)>, x) & \text{otherwise} \end{cases} \]

The idea is that if \( w \) witnesses \( (B \vee C) \vdash \varphi(\vec{A}^*) \) 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 \( \forall \Delta_i \); in the latter case, \( h \) is used. This can easily be formalized in \( S_{2^1}^1 \), so

\[ S_{2^1}^1 \vdash \text{Witness}_{\Sigma^P_1}(f(w, x), x). \]

Now \( f \) is a \( \Sigma^P_1 \)-function since \( g \) and \( h \) are and by Proposition 4. Also, \( f \) is \( \Sigma^P_1 \)-definable by \( S_{2^1}^1 \) since \( g \) and \( h \) are and since \( \text{Witness}_{\Sigma^P_1} \) is a \( \Delta^P_1 \)-predicate.

Case (4): \((\exists x \subseteq [1]: \text{left})\). Suppose the last inference of \( P \) is

\[
\frac{\varphi(x, x, \beta(1, w), \beta(2, w), \alpha) \rightarrow \Delta}{(\exists x \subseteq [1]: \beta(x), \alpha) \rightarrow \Delta}
\]

Of course, \( a \) is an eigenvariable and must not appear in the lower sequent. Let \( D \) be the formula \( \varphi(x, x, \beta(1, w), \beta(2, w), \alpha) \), and let \( E \) be \((\exists x \subseteq [1]: \beta(x), \alpha) \). By the induction hypothesis, there is a \( \Sigma^P_1 \)-definable function \( g(\vec{A}^*) \) such that

\[ S_{2^1}^1 \vdash \text{Witness}_{\Sigma^P_1}(g(w, x), x, a). \]

(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 \( \Delta \).)

First consider the case where \((\exists x \subseteq [1]: \beta) \) is not in \( \Sigma^P_1 \). Define the function \( h \) by

\[ h(w) = <\beta(1, \beta(1, w)), \beta(2, w)>. \]

By the definition of \( \text{Witness} \) we have
§5.2 The Main Proof

\[ S^1_1: \text{Witness}^0 \left( w, \exists \right) \supset \text{Witness}^0 \left( \beta(h(w), \exists, \beta(1, 1, w)) \right). \]

So define \( f \) by

\[ f(w, \exists) = g(h(w), \exists, \beta(1, 1, w)). \]

Thus \( f \) is a \( \Omega^1 \)-function, \( f \) is \( \Sigma^1 \)-definable by \( S^1_2 \) and

\[ S^2_2: \text{Witness}^0 \left( w, \exists \right) \supset \text{Witness}^0 \left( \beta(h(w), \exists, \beta(1, 1, w)) \right). \]

The case where \( (\exists x \leq t) B(x) \in \Sigma^1 \) is even easier. We now let

\[ h(w) = \langle 0, 0, \beta(2, w), \beta(1, 1, w) \rangle \]

and

\[ f(w, \exists) = g(h(w), \exists, \beta(1, 1, w)). \]

Note \( f \in \Omega^1 \) since \( (\mu x \leq t) B(x) \) can be computed either by using a binary search or, when \( (\exists x \leq t) \) is a sharply bounded quantifier, by an exhaustive search.

Case (3): \( (\forall z \leq x): \) Suppose the last inference of \( P \) is

\[ B(x) \Gamma^* \supset \Delta \]

\[ \forall z \leq t : (\forall z \leq x) B(x) \Gamma^* \supset \Delta \]

We shall assume that \( s \leq t \) is in \( \Gamma \) (a similar argument works for \( s \leq t \) in \( \Pi \)). Let \( D \) be the formula \( B(x) \left( \exists \right) \beta(1, 1, w) \), and let \( E \) be \( s \leq t \langle (\forall z \leq x) B(x) \rangle \beta(1, 1, w) \). The induction hypothesis is that there is a \( \Sigma^1 \)-definable function \( g \) in \( \Omega^1 \) such that

\[ S^1_2: \text{Witness}^0 \left( w, \exists \right) \supset \text{Witness}^0 \left( g(h(w), \exists, \beta(1, 1, w)) \right). \]

First consider the case where \( (\forall z \leq x) B(x) \) is not in \( \Sigma^1 \). Then \( (\forall z \leq x) \) must be a sharply bounded quantifier with \( t = |r| \) for some term \( r \). Define the function \( h \) by

\[ h(w, \exists) = \langle \beta(1, 1, w), \beta(1, 1, w), \beta(1, 1, w) \rangle. \]

By the definition of \( \text{Witness} \), we have

\[ S^2_2: \text{Witness}^0 \left( w, \exists \right) \supset \text{Witness}^0 \left( g(h(w), \exists, \beta(1, 1, w)) \right). \]

So define \( f(w, \exists) = g(h(w), \exists, \beta(1, 1, w)) \). It is straightforward to see that \( f \) satisfies the desired conditions of Theorem 3.
The case where $(\forall x \leq t) B(x) = \Sigma^a_{\lambda} \cup \Pi^a_{\lambda}^1$ is easier. We now set $h(w, z)$ equal to $<\emptyset, \emptyset, \emptyset, \emptyset>$ and otherwise proceed as before.

Case (6): $\langle \cdot, \cdot \rangle$ (left) and $\langle \cdot, \cdot \rangle$ (right). We omit these cases see (v: left) and (v: right).

Case (7): $\langle v, \cdot \rangle$. Suppose the last inference of $P$ is

$$
\frac{\Gamma \rightarrow B \cdot \Delta^*}{\Gamma \rightarrow B \cdot C \cdot \Delta^*}
$$

Let $D$ be the formula $Bv(\forall \Delta^*)$. By the induction hypothesis, there is a $\Sigma^b_{\lambda}$-definable function $g$ in $\Omega^b$ such that

$$
S^I_2 \vdash \text{Witness}^b_{K}(w, z) \supset \text{Witness}^b_{K}(g(v, z), z).
$$

Define $h$ by

$$
h(w) = \langle <\emptyset(1, w), \emptyset>, \emptyset(2, w) >
$$

and let $f(w, z) = h(g(v, z))$. Then it is easy to see that $f$ satisfies all the desired conditions.

Case (8): $\langle \lambda, \cdot \rangle$. Suppose the last inference of $P$ is

$$
\frac{\Gamma \rightarrow B \cdot \Delta^* \quad \Gamma \rightarrow C \cdot \Delta^*}{\Gamma \rightarrow B \cdot C \cdot \Delta^*}
$$

Let $D$ be the formula $Bv(\forall \Delta^*)$, let $E$ be $Cv(\forall \Delta^*)$ and let $F$ be $(B \land C)v(\forall \Delta^*)$. The induction hypothesis is that there are $\Omega^b_{\lambda}$-functions $g$ and $h$ which are $\Sigma^b_{\lambda}$-definable by $S^I_2$ such that

$$
S^I_2 \vdash \text{Witness}^b_{K}(w, z) \supset \text{Witness}^b_{K}(g(v, z), z)
$$

and

$$
S^I_2 \vdash \text{Witness}^b_{K}(w, z) \supset \text{Witness}^b_{K}(h(v, z), z).
$$

We define the function $k$ as

$$
k(v, w, z) = \begin{cases} 
v & \text{if } \text{Witness}^b_{K}(v, z) \\
w & \text{otherwise} \end{cases}
$$
By Proposition 4, $k$ is a $\Omega^\ast$-function; also, $k$ is $\Sigma^k_1$-definable by $S^k_2$ since $\text{Witness}^F_\Delta^\ast$ is a $\Delta^k_1$-formula. Now define $f$ by

$$f(w,\zeta) = \langle \langle \beta(1, g(w, \zeta)), \beta(1, h(w, \zeta)), k(\beta(2, g(w, \zeta)), \beta(2, h(w, \zeta)), \zeta) \rangle \rangle.$$ 

Clearly $f$ is $\Sigma^k_1$-definable by $S^k_2$ and is in $\Omega^\ast$, since $g$, $h$, and $k$ are. Also, it is easy to see that

$$S^k_2 \vdash \text{Witness}^F_\Delta^\ast \psi \rightarrow \text{Witness}^F_\Delta^\ast (f(w, \zeta), \zeta).$$

Case (9): ($\exists \leq \text{right}$). Suppose the last inference of $P$ is

$$\xi \leq t, \Gamma \vdash B(s) \Delta^\ast \\
\xi \xi \leq t, \Gamma \vdash (\exists x \leq t) B(x, \Delta^\ast)$$

We shall assume that $s \leq t$ is in $\Gamma$ (a similar argument works for $s \leq t$ in $\Pi$). Let $D$ be the formula $B(x) \psi (\forall \Delta^\ast)$, let $E$ be $s \leq t \forall (\forall \Delta^\ast)$ and let $F$ be $(\exists x \leq t) B(x) \psi (\forall \Delta^\ast)$. The induction hypothesis is that there is a $\Omega^\ast$-function $g$ which is $\Sigma^k_1$-definable in $S^k_2$ such that

$$S^k_2 \vdash \text{Witness}^F_\Delta^\ast (w, \zeta) \rightarrow \text{Witness}^F_\Delta^\ast (g(w, \zeta), \zeta).$$

By the definition of $\text{Witness}$,$$
S^k_2 \vdash \text{Witness}^F_\Delta^\ast (w, \zeta) \rightarrow \text{Witness}^F_\Delta^\ast (g(w, \zeta), \zeta).$$

So define $f$ to be

$$f(w, \zeta) = \langle \langle e(\zeta), \beta(1, g(2, w, \zeta)), \beta(2, g(2, w, \zeta)) \rangle \rangle.$$ 

Then $f$ is $\Sigma^k_1$-definable by $S^k_2$, $f$ is a $\Omega^\ast$-function and

$$S^k_2 \vdash \text{Witness}^F_\Delta^\ast (w, \zeta) \rightarrow \text{Witness}^F_\Delta^\ast (f(w, \zeta), \zeta).$$

Case (10): ($\forall \leq \text{right}$). Suppose the last inference of $P$ is

$$\xi \leq t, \Gamma \vdash B(s) \Delta^\ast \\
\Gamma \rightarrow (\forall x \leq t) B(x, \Delta^\ast)$$

where $a$ is the eigenvariable and does not appear free in the lower sequent. Let $D$ be the
formula $a \leq tr(\forall \exists !)$, let $E$ be $B(a)v(\forall \Delta^*)$ and let $F(\exists !, t)$ be $(\forall s \leq t)B(x)v(\forall \Delta^*)$. The induction hypothesis is that there is a $\Sigma^F_{\leq}$-function $g$ such that

$$S^F_{\leq} \vdash \text{Witness}_{\Sigma^F_{\leq}}(g, w, \exists !, t), \exists !, t).$$

First, consider the case where $(\forall s \leq t)B(a)$ is not in $\Sigma^F_{\leq} \cap \Pi^F_{\leq}$. So $(\forall s \leq t)$ is sharply bounded with $r = |r|$, for some term $r$. We define the function $k$ by

$$k(w, w, \exists !) = \begin{cases} v & \text{if Witness}_{\Delta^*}(v, \exists !) \\ w & \text{otherwise} \end{cases}$$

We define $f$ by the following limited iteration scheme:

$$p(w, \exists !, 0) = \langle < \emptyset, g(w, \exists !, 0) >, \beta(2, g(w, \exists !, 0)) >$$

$$p(w, \exists !, m+1) = \langle < \emptyset, p(w, \exists !, m) >, \beta(1, g(w, \exists !, m+1)), \beta(2, p(w, \exists !, m)) >$$

$$f(w, \exists !) = p(< 0, w, \exists !, |r|)$$

By Proposition 4, $k \in \Sigma^F_{\leq}$ and hence $f \in \Delta^*$. It is straightforward to see that

$$S^F_{\leq} \vdash \text{Witness}_{\Delta^*}(f, w, \exists !, t)$$

and

$$S^F_{\leq} \vdash \text{Witness}_{\Delta^*}(\exists !, d) \Rightarrow \text{Witness}_{\Delta^*}(p(w, \exists !, d), \exists !, d)$$

$$\Rightarrow \text{Witness}_{\Delta^*}(p(w, \exists !, d+1), \exists !, d+1).$$

It follows by $\Sigma^F_{\leq} \cdot \text{LIND}$ that

$$S^F_{\leq} \vdash \text{Witness}_{\Sigma^F_{\leq}}(f, w, \exists !, t).$$

Hence,

$$S^F_{\leq} \vdash \text{Witness}_{\Delta^*}(f, w, \exists !, t)$$

which is what we needed to show.

Second, consider the case where $(\forall s \leq t)B(a)$ is in $\Sigma^F_{\leq} \cap \Pi^F_{\leq}$. If $A(x, a)$ is any one of the formulae $a \leq t, B(a)$ or $(\forall s \leq t)B(a)$, then Witness$_{\Delta^*}(w, \exists !, a)$ is defined to be equivalent to $A(\exists !, a)$ itself. Let $h(w, \exists !)$ be the $\Delta^F$-function $(\forall s \leq t)B(x)$ and let
\[ f(w, \tau) = g(<0, w, \tau, h(w, \tau)) \]. Then \( f \) satisfies the desired conditions.

Case (11): (Cut). Suppose the last inference of \( P \) is

\[
\begin{align*}
\Gamma & \rightarrow \Delta, B, \gamma \rightarrow \Delta \\
\Gamma & \rightarrow \Delta
\end{align*}
\]

Since \( P \) is free cut free, \( B \) must be a \( \Sigma^A_1 \)-formula. Let \( D \) be the formula \( B \land (\forall \Delta) \) and let \( E \) be \( B \land (\forall \Gamma) \). The induction hypothesis is that there are \( \Pi^A_1 \)-functions \( g \) and \( h \) which are \( \Sigma^A_1 \)-defined by \( S^\xi_2 \) such that

\[
S^\xi_2 \vdash \text{Witness}_{\xi_2}^\alpha (w, \tau, \tilde{\tau}) \rightarrow \text{Witness}_{\tilde{\xi}}^\alpha (g(w, \tau), \tilde{\tau})
\]

and

\[
S^\xi_2 \vdash \text{Witness}_{\xi_2}^\alpha (w, \tau, \tilde{\tau}) \rightarrow \text{Witness}_{\xi_2}^\alpha (h(w, \tau), \tilde{\tau}).
\]

We define the function \( f \) as

\[
f(w, \tau) = \begin{cases} 
\delta(2, g(w, \tau)) & \text{if Witness}_{\xi_2}^\alpha (g(2, g(w, \tau)), \tilde{\tau}) \\
h(<\beta[1, g(w, \tau), w, \tau]) & \text{otherwise}
\end{cases}
\]

By Proposition 4, \( f \in \Pi^A_1 \), and since \( \text{Witness}_{\xi_2}^\alpha \) is \( \Delta^A_1 \) with respect to \( S^\xi_2 \), \( f \) is \( \Sigma^A_1 \)-defined by \( S^\xi_2 \). Also, it is easy to see that

\[
S^\xi_2 \vdash \text{Witness}_{\xi_2}^\alpha (w, \tau, \tilde{\tau}) \rightarrow \text{Witness}_{\xi_2}^\alpha (f(w, \tau), \tilde{\tau}).
\]

Case (12): (\( \Sigma^A_1 \)-PIND). Suppose the last inference of \( P \) is

\[
\frac{B(\{s\} \uparrow), \Delta^* \rightarrow B(\alpha), \Delta^*}{B(0), \Delta^* \rightarrow B(\tau), \Delta^*}
\]

where \( \alpha \) is an eigenvariable and must not appear in the lower sequent. We shall only consider the case where \( B(0) \) is in \( \Gamma \) and \( B(\alpha) \) is in \( \Delta \). (If this is not the case, then \( B \in \Sigma^A_1 \land \Pi^A_1 \), and the argument is much simpler.)

The general idea of the argument for Case (12) is to treat the \( \Sigma^A_1 \)-PIND inference as if it were \(|\tau| = 1\) cuts. So, in effect, Case (12) is handled by iterating the method of Case (11).

Let \( \mathcal{A} \) be the formula \( B(\{s\}) \land (\forall \Delta^*) \), let \( \mathcal{B} \) be \( B(\alpha) \lor (\forall \Delta^*) \), let \( F \) be \( B(0) \lor (\forall \Delta^*) \) and let \( \lambda (\tau, d) \) be \( B(d) \lor (\forall \Delta^*) \). The induction hypothesis is that there is a \( \Pi^A_2 \)-function \( g \).
such that

$$S^*_l \vdash \text{Witness}_{p^*}(w, \mathcal{F}, \alpha) \supset \text{Witness}_{p^*}(g(w, \mathcal{F}, \alpha), \mathcal{F}, \alpha).$$

We define $\Omega^*_l$-functions $k$ and $h$ by

$$k(v, w, \mathcal{F}) = \begin{cases} v & \text{if } \text{Witness}_{\Omega^*_l}(v, \mathcal{F}) \\ w & \text{otherwise} \end{cases}$$

and

$$h(s, w, \mathcal{F}, \alpha) = g(\langle \emptyset(1, v), \emptyset(2, w), \mathcal{F}, \alpha \rangle).$$

By Proposition 3(c) there is a term $t_A$ and a $\Omega^*_l$-function $q$ which is $\Sigma^l_A$-definable in $S^*_l$ such that

$$S^*_l \vdash \text{Witness}_{\Omega^*_l}(w, \mathcal{F}, \alpha) \supset \text{Witness}_{\Omega^*_l}(q(w, \mathcal{F}, \alpha), \mathcal{F}, \alpha) \supset q(w) \leq t_A(\mathcal{F}, \alpha).$$

Now define $f^*$ by the following limited iteration scheme:

$$p(w, \mathcal{F}, 0) = q(\emptyset(1, w), 0)$$

$$p(w, \mathcal{F}, m+1) = q(\emptyset(1, h(p(w, \mathcal{F}, m), \mathcal{F}, \alpha, \text{MSP}(t_1[t_1 \rightarrow (m+1)]))))$$

$$k(\emptyset(2, p(w, \mathcal{F}, m)), \emptyset(2, h(p(w, \mathcal{F}, m), \mathcal{F}, \alpha, \text{MSP}(t_1[t_1 \rightarrow (m+1)]))), \mathcal{F}, \alpha)$$

$$f^*(w, \mathcal{F}, u) = p(w, \mathcal{F}, [u]).$$

This is a valid limited iteration definition since the use of the function $q$ gives a provable polynomial space bound on $p$; namely, $p(w, \mathcal{F}, m) \leq \sigma(t_A(\mathcal{F}, \alpha))$. Thus $f^*$ is a $\Omega^*_l$-function which is $\Sigma^l_A$-definable in $S^*_l$.

Now it is easy to see that

$$S^*_l \vdash \text{Witness}_{\Omega^*_l}(p^*(w, \mathcal{F}, \alpha), \alpha) \supset \text{Witness}_{\Omega^*_l}(f^*(w, \mathcal{F}, \alpha), \alpha)$$

and

$$S^*_l \vdash \text{Witness}_{\Omega^*_l}(p^*(w, \mathcal{F}, \alpha), \text{Witness}_{\Omega^*_l}(f^*(w, \mathcal{F}, u), \alpha, \text{MSP}(t_1[t_1 \rightarrow [u]])) \supset \text{Witness}_{\Omega^*_l}(f^*(w, \mathcal{F}, u), \alpha, \text{MSP}(t_1[t_1 \rightarrow [u]])).$$

So by $\Sigma^l_A$-PIND with respect to $u$. 

\[ S^2_1 \vdash \text{Witness}_{\Sigma^2_1}(w, \exists t) \supset \text{Witness}_{\Sigma^2_1}(f^*(w, \exists t), \exists t). \]

So define \( f(w, \exists t) = f^*(w, \exists t) \) and we are done.

Case (3)\( \xi \): (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 \( \Sigma^i_1 \)-formula and that \( S^2_1 \vdash (\forall \exists y) A(x, y) \). Then there is a term \( t \), a \( \Sigma^1_1 \)-formula \( B \) and a function \( f \in \Omega^i_1 \) such that

1. \( S^2_1 \vdash (\forall \exists y)(\exists \exists z \leq t) B(x, y) \)
2. \( S^2_1 \vdash (\forall \exists y)(\exists \exists z \leq t) B(x, y) \supset A(x, y) \)
3. \( S^2_1 \vdash (\forall \exists y)(\exists \exists z \leq t) B(x, y) \supset B(x, z) \supset y = z \)
4. For all \( \bar{\Pi} \in B(\bar{\Pi}, f(\bar{\Pi})) \)

Conversely, if \( f \in \Omega^i_1 \), then there is a term \( t \) and a \( \Sigma^1_1 \)-formula \( B \) such that (1), (3) and (4) hold.

**Corollary 7:** Let \( i \geq 1 \). A function \( f \) is \( \Sigma^i_1 \)-definable in \( S^2_1 \) iff \( f \in \Omega^i_1 \).

For the special case \( i = 1 \), we have

**Corollary 8:** The \( \Sigma^1_1 \)-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 \( \Sigma^i_1 \)-formula, \( B \) is a \( \Pi^i_1 \)-formula and that \( S^2_1 \vdash A(\bar{a}) \supset B(\bar{a}) \). Then there is a predicate \( Q \in \Delta^i_1 \) such that for all \( \bar{a} \),

\[ Q(\bar{a}) \iff N \models A(\bar{a}) \iff N \models B(\bar{a}). \]
Conversely, if \( q \in \Delta^1 \), 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

\[
f(x) = \begin{cases} 
0 & \text{if } A(x) \\
1 & \text{otherwise}
\end{cases}
\]

is \( \Sigma^1 \)-definable in \( S^1_2 \) by the equation

\[
f(x) = y \iff [y = 0 \land A(x)] \lor [y = 1 \land \neg B(x)].
\]

Thus \( f \in \Sigma^1_{2} \) and hence \( A \) represents a predicate in \( P \).

Recall that in Chapter 1 we characterized the \( NP \) predicates as those expressible by \( \Sigma^1_{2} \)-formulae and the \( co-NP \) predicates as those expressible by \( \Pi^1_{2} \)-formulae. Hence in the case \( r = 1 \), Theorem 9 becomes:

**Corollary 10:** Let \( A(x) \) be a formula such that \( S^1_2 \) proves \( A \) is equivalent to a \( \Sigma^1_{2} \)- and to a \( \Pi^1_{2} \)-formula (i.e., \( S^1_2 \) proves that \( \lambda \in NP \cap co-NP \)). Then \( A(\tilde{x}) \) is a polynomial time predicate (i.e., \( A \) is in \( P \)).

So any predicate which is \( S^1_2 \)-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 \( S^1_2 \) in the following way:

**Definition:** Let \( k \geq 1 \) and let \( p(n_1, \ldots, n_k) \) be a suitable polynomial. For each \( j \geq 0 \), \( \eta^j_A \) is a \( k \)-ary function symbol. The **bundling axiom** for \( \eta^j_A \) is

\[
[\eta^j_A(\bar{a})] = p(|\bar{a}|, \ldots, |\bar{a}|).
\]

**Definition:** Let \( \eta^0_{A_1}, \ldots, \eta^k_{A_k} \) be a sequence of function symbols. We write \( \bar{y} \) as an abbreviation for that sequence. The theory \( S^1_2(\bar{y}) \) is defined to be the theory with the language of Bounded Arithmetic plus the symbols \( \eta^0_{A_1}, \ldots, \eta^k_{A_k} \), and with the following axioms:

<table>
<thead>
<tr>
<th>Axiom</th>
</tr>
</thead>
<tbody>
<tr>
<td>[ \eta^j_A(\bar{a}) = p(</td>
</tr>
<tr>
<td>[ \forall \bar{y} \forall \bar{a} \exists \bar{y}' \forall \bar{a}' [ \eta^j_A(\bar{a}) = p(</td>
</tr>
<tr>
<td>[ \forall \bar{y} \exists \bar{y}' \forall \bar{a} \exists \bar{a}' [ \eta^j_A(\bar{a}) = p(</td>
</tr>
</tbody>
</table>

\[ \forall \bar{y} \forall \bar{a} \exists \bar{b} \forall \bar{c} [ \eta^j_A(\bar{a}) = p(|\bar{a}|, \ldots, |\bar{a}|) ] \]
(1) the **BASIC** axioms

(2) for each \(1 \leq t \leq n\), the bounding axiom: \([\eta_t^\omega, t] \leq p, \eta_t\)

(3) the \(\Sigma_t^k(\eta)\)-**PIND** axioms.

Recall that in Chapter 1 \(\omega^t\) was defined to be the set of \(k\)-ary functions with growth rate bounded by the polynomial \(p\).

**Definition:** If \(\eta_t^\omega\) is a function symbol then the **function space associated with** \(\eta_t^\omega\) is \(\omega^t\).

We can relativize Theorem 1 as follows:

**Theorem 11:** Let \(\vec{\eta}\) be a vector of function symbols and let \(\vec{\omega} = \omega_{a_1}^{\eta_1}, \ldots, \omega_{a_n}^{\eta_n}\) be the vector of function spaces associated with the \(\eta_t\)'s. Let \(i \geq 1\) be fixed.

Suppose \(S_2^\omega(\vec{\eta}) = (\forall \bar{y})(\exists \bar{z}) A(\bar{x}, \bar{y})\) where \(A\) is a \(\Sigma_1^k(\vec{\eta})\)-formula and \(\bar{x}\) and \(\bar{y}\) are the only free variables in \(A(\bar{x}, \bar{y})\). Then there is a term \(t(\bar{x}, \bar{y})\), a \(\Sigma_1^k(\vec{\eta})\)-formula \(B\) and a functional \(g\) in \(\Sigma_t^k(\omega)\) so that

1. \(S_2^\omega(\vec{\eta}) = (\forall \bar{y})(\forall \bar{z})(B(\bar{x}, \bar{y}, \bar{z}) \supset A(\bar{x}, \bar{y}))\)
2. \(S_2^\omega(\vec{\eta}) = (\forall \bar{y})(\forall \bar{z})(y \in g(B(\bar{x}, \bar{y}, \bar{z})) \supset g(y) = z)\)
3. \(S_2^\omega(\vec{\eta}) = (\forall \bar{y})(\exists \bar{z} \leq t(\bar{x}, \bar{y}) B(\bar{x}, \bar{y}, \bar{z}))\)
4. For all \(\vec{\alpha} \in \mathbb{N}^n\) and all oracles \(\Omega_1, \ldots, \Omega_n\) with \(\Omega_i \in \omega_{a_i}^\omega\) for all \(1 \leq i \leq n\),

\[ (\mathbb{N}, \Omega) \models B(\bar{x}, g(\bar{y}, \Omega)). \]

**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 \(\Delta_1^k\)-formulae with respect to \(S_2^\omega(\vec{\eta})\) and \(g\) is a suitable polynomial and that

\[ S_2^\omega(\vec{\eta}) = (\forall \bar{y})(\exists \bar{z} < 2^{\omega(\alpha)}) B(\bar{x}, \bar{y}) \supset (\forall \bar{z}) A(x, \bar{y}). \]

Then there is a functional \(g \in \Omega^\omega_{\omega(\alpha)}\) such that whenever \(\Omega \in \omega_t^\omega\),

\[ (\mathbb{N}, \Omega) \models (\forall x) A(\bar{x}, g(\bar{x}, \Omega)). \]
Recall that the condition that \( g \in T(\Omega^f) \) means that \( g \) can be computed by a deterministic, polynomial-time, oracle Turing machine \( M_g \) where the function oracle \( \Omega \) used by \( M_g \) is required to satisfy \( \Omega(x) \leq g(|x|) \) for all \( x \).

**Proof:** By the hypothesis of the corollary,

\[
S^L_2(\eta^f_{\Omega}):(\forall x)(A(x,y) \equiv (\exists y)(B(y,\eta^f_{\Omega}(y))).
\]

So,

\[
S^L_2(\eta^f_{\Omega}):(\forall y)(A(x,y) \equiv \neg B(y,\eta^f_{\Omega}(y))).
\]

Then, by Theorem 11 there is a \( \psi \in T(\Omega^f) \) such that, for all \( x \) and all \( \Omega \in \omega^f \), \( \psi(x,\Omega) \) is equal to either \( \eta \) such that \( A(x,\eta) \) holds or \( \eta \) such that \( B(y,\eta) \) fails.

Q.E.D. \( \Box \)

**Definition:** Let \( f \) be a unary function symbol. Then \( PHP(f) \) is an abbreviation for the formula

\[
\forall x(y < 2a \rightarrow f(y) < a) \land (\exists z < 2a)(f(y) = f(z) \land y \neq z).
\]

So \( PHP(f) \) expresses a pigeon hole principle stating that \( 2 \cdot a \) pigeons cannot sit in \( a \) holes. Note that \( a \) appears as a free variable in \( PHP(f) \).

**Corollary 15:** \( S^L_2(f) = PHP(f) \).

Of course, \( S^L_2(f) \) means the theory extending \( S^L_2 \) with the new function symbol \( f \) and the \( \Sigma^L_1(f) \)-PIND axioms.

**Proof:** Suppose the corollary is false, then let \( \eta \) be the polynomial \( \eta(n) = n \). Then

\[
S^L_2(\eta^f_{\Omega}): PHP(\eta^f_{\Omega}).
\]

Hence,

\[
S^L_2(\eta^f_{\Omega}): a \neq 0 \land (\exists y < 2a)(\exists z < 2a) (\eta^f_{\Omega}(y) = \eta^f_{\Omega}(z) \land y \neq z).
\]

So by Theorem 11 there is an \( f \in T(\Omega^f) \) such that for all \( a \in \mathbb{N} \) and all oracles \( \Omega \in \omega^f \),

\[
f(a,\Omega) = \langle y, z \rangle \text{ where } y \neq z \text{ and } y \text{ satisfies the above condition.}
\]

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 \( \Omega_0 \) so that the following conditions hold:

If $M(x_0, \Omega_0)$ first queries its oracle for the value of $\Omega_d(m)$ on the $n$-th step where $m < 2^n$, then set $\Omega_d(m)$ to be equal to the greatest number $j \leq \min(m, x_0 - 2)$ such that no earlier oracle query of $M(x_0, \Omega_0)$ yielded the answer $j$. Such a $j$ will always exist.

If $M(x_0, \Omega_0) = \langle y_0, r_0 \rangle$ and if $\Omega_d(y_0)$ and/or $\Omega_d(z_0)$ have not yet been defined, set $\Omega_d(y_0) = x_0 - 1$ and/or set $\Omega_d(z_0) = x_0 - 2$.

For all other values of $m$, set $\Omega_d(m) = 0$.

Q.E.D.

Corollary 13 states that $S_2^f(f)$ can not prove the pigeon hole principle $PHP(f)$. On the other hand, Alex Wilkie [30] showed that $S_2^f(f)$ can prove $PHP(f)$. Examining Wilkie’s proof closely yields the following theorem:

**Theorem 14**: (Wilkie [30]). $T_2^f(f) \vdash PHP(f)$

Combining Wilkie’s theorem and Corollary 13 gives

**Corollary 15**: $T_2^f(f)$ is not equivalent to $S_2^f(f)$.

It is an open question whether $S_2^f$ is equivalent to $T_2^f$ or even if $S_2^f$ 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 scheme which allows function symbols to be introduced for each polynomial time function and an induction schema which is essentially equivalent to the FIND axioms applied to open formulae of PV.

Our earlier results have shown that $S^1_2$ can $\Sigma^1_1$-define precisely the polynomial time functions. Thus it is not too surprising that $S^1_2$ and PV are closely related. We shall see below that, after making allowances for the fact that they have different languages, $S^1_2$ and PV have the same $\Sigma^1_1$-formulae as theorems.

6.1. Preliminaries for PV and PV1.

Like $S^1_2$, 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 $d_kd_{k-1}\cdots d_0$ where $n=\sum_{i=0}^{k} 2^id_i$ 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

$$s_i(d_kd_{k-1}\cdots d_0) = d_kd_{k-1}\cdots d_{i+1}$$

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

$$f(0,y) = g(y)$$
$$f(s_1(x),y) = h_i(x,y,f(x,y))$$

provided that PV proves
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  

\[
\text{PROOF}(m,n) = \begin{cases} 
1 & \text{if } m \text{ is the Gödel number of an equation and} \\
0 & \text{otherwise.} 
\end{cases}
\]

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 \( S, +, \cdot, \# \), \( \text{\#}[x] \) and \( ||x|| \) as well as functions for handling sequences; namely, \( S(1,w) \), the pairing function

\[
(w_1, w_2) \mapsto \langle w_1, w_2 \rangle
\]

and the sequence extension function

\[
< a_1, \ldots, a_n > \cdot a_{n+1} = < a_1, \ldots, a_{n+1} >.
\]

(Our definition for \( \cdot \) conflicts with the notation in [6]. Our function \( \cdot \) 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 \( LE(x,y) = 0 \) 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 \(|x|_d\), \( PV \) can define the dyadic length function \(|x|_2\) 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:

\[
\begin{align*}
\text{NOT}(x) &= \begin{cases} 
0 & \text{if } x \neq 0 \\
1 & \text{if } x = 0 
\end{cases} \\
\text{AND}(x,y) &= \begin{cases} 
0 & \text{if } x = 0 \text{ or } y = 0 \\
1 & \text{otherwise} 
\end{cases} \\
\text{OR}(x,y) &= \text{NOT(AND(\text{NOT}(x),\text{NOT}(y)))} 
\end{align*}
\]

We can, in effect, use sharply bounded quantifiers in PV by introducing new function symbols which have a similar effect:

**Definition:** Let \(P(x,\bar{x})\) and \(F(\bar{x})\) be PV-function symbols. Then

\[
Q(\bar{x}) = (\exists x \leq |F(\bar{x})|)P(x,\bar{x})
\]

is a PV-function symbol so that

\[
Q(\bar{x}) = \begin{cases} 
1 & \text{if } (\forall x \leq |F(\bar{x})|)\{(x \land P(x,\bar{x}))\} \\
0 & \text{otherwise} 
\end{cases}
\]

\(Q(\bar{x})\) is defined by the following limited iteration on notation scheme:

\[
\begin{align*}
f(0,\bar{x},x) &= \text{NOT(\text{NOT}(P(0,\bar{x})))} \\
f(s_1(y),x,\bar{x}) &= \text{AND}(f(y,\bar{x},x)\land \text{OR}(P[|s_1(y)|,\bar{x}],\text{LE}(x,|s_1|))) \\
Q(\bar{x}) &= f(s_1(F(\bar{x})),\bar{x},|F(\bar{x})|)
\end{align*}
\]

Also, \((\exists x \leq |F(\bar{x})|)P(x,\bar{x}))\) is defined to be the PV-function symbol \(\text{NOT}(\text{NOT}(\forall x \leq |F(\bar{x})|)P(x,\bar{x}))\).
Proposition 1: Let $G(x,y)$ be any $PV$-function symbol. Then there is a $PV$-function symbol $F(x,y)$ so that

$$PV \vdash \beta(0,F(x,y))-|y|+10w(|y|)(\forall z \leq |y|)(G(x,z)=\beta z+1,F(x,y))).$$

Proposition 1 states that $PV$ satisfies an analogue of the $\Delta_1^1$-replacement property, in that the value of $F(x,y)$ is $< G(x,0), \ldots, G(x,|y|) >$.

Proof: Let $H(x,y,z)$ be the $PV$-function symbol defined by the following limited iteration on notation:

$$H(x,y,0) = < G(x,0) >$$

$$H(x,y,z(w)) = \left\{ \begin{array}{ll}
H_0 & if \; |w| < |y| \\
H(x,y,w) & otherwise
\end{array} \right.$$  

Now set $F(x,y)=H(x,y,p(y))$.

6.2. $S_2^1$ 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 $\Sigma_1^1$-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 $\Sigma_1^1$-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 $\Sigma_1^1$-defining 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 $\Delta_1^1$-IND 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 $\Sigma_1^i(L_{PV})$-**PIND** axioms,
3. The following axioms defining the initial function symbols of PV (compare with [6]):
   
   
   $s(0) = 2x + 1$
   
   $s(s(x)) = 2x + 2$
   
   $TR(0) = 0$
   
   $TR(s(s(x))) = x$
   
   $z \cdot 0 = x$
   
   $z \cdot s(y) = s(z \cdot y)$
   
   $z \cdot x = x \cdot (y \cdot z)$
   
   $\text{LESS}(x, 0) = x$
   
   $\text{LESS}(x, s(y)) = TR(\text{LESS}(x, y))$

   where $i = 1, 2$ and where we are using $\cdot$ to denote Cook’s function $\ast$ since $\ast$ 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

   $f(0, y) = y(y)$

   $f(s(x), y) = h(x, y, f(x, y))$

**Proposition 8:** $S_2^1(L_{PV})$ 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 $\Sigma_1^i$-defining axiom $f(x) = \iff A_f(x, y)$ 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 $\Sigma_1^i$-defining axiom $f'(x) = y \iff A_f(x, y)$. 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 $(\forall x)(f'(x) = f(x))$, 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 \( S_2^4(PV) \vdash t = u \) if \( PV \vdash t = u \).

One direction of this theorem is immediate from our remark above that \( S_2^4(L_{PV}) \) 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 \( \Sigma_1^1 \)-Formula.

For the sake of avoiding excessive subscripts, we use \( \Sigma_1^1(PV) \) and \( \Pi_1^1(PV) \) as synonyms for \( \Sigma_1^1(L_{PV}) \) and \( \Pi_1^1(L_{PV}) \) from now on.

In order to handle \( \Sigma_1^1(PV) \)-formulæ in the theory \( PV \), we need a way for \( PV \) to assert that a given \( \Sigma_1^1(PV) \)-formula is true.

Definition: Let \( A \) be a \( \Sigma_1^1(PV) \)-formula and \( \mathbf{a} \) be a vector of \( k \) free variables containing all the free variables of \( A \). \( \text{WITNESS}_A^k \) is a \((k+1)\)-ary function symbol of \( PV \) defined by induction on the complexity of \( A \) as follows:

1. If \( A \) is atomic,
   \[
   \text{WITNESS}_A^k(w, \mathbf{a}) = \begin{cases} 
   1 & \text{if } A(\mathbf{a}) \\
   0 & \text{otherwise}
   \end{cases}
   \]

2. If \( A \) is \( B \lor C \),
   \[
   \text{WITNESS}_A^k(w, \mathbf{a}) = \text{AND}(\text{WITNESS}_B^k(\beta(1, w), \mathbf{a}), \text{WITNESS}_C^k(\beta(2, w), \mathbf{a}))
   \]

3. If \( A \) is \( B \land C \),
   \[
   \text{WITNESS}_A^k(w, \mathbf{a}) = \text{OR}(\text{WITNESS}_B^k(\beta(1, w), \mathbf{a}), \text{WITNESS}_C^k(\beta(2, w), \mathbf{a}))
   \]

4. If \( A \) is \((\forall x \leq t)\theta(x, \mathbf{a})\) where \( \theta(0, \mathbf{a}) \in \Sigma_1^1(PV) \), then
   \[
   \text{WITNESS}_A^k(w, \mathbf{a}) = (\forall x \leq t)\text{WITNESS}_A^k(\beta(x+1, w), x, \mathbf{a})
   \]

5. If \( A \) is \((\exists x \leq t)B(x, \mathbf{a})\) where \( B \in \Sigma_1^1(PV) \), then
   \[
   \text{WITNESS}_A^k(w, \mathbf{a}) = \text{WITNESS}_B^k(\beta(2, w), \beta(1, w), \mathbf{a}) \land \beta(1, w) \leq t
   \]

6. If \( A \) is \( \neg B \) then we transform \( A \) by logical operations so that Cases (1)-(5) apply. Specifically, if \( A \) is \( \neg(\neg B) \), \( \neg(B \lor C) \), \( \neg(\forall x \leq t)B \) or \( \neg(\exists x \leq t)B \), then let \( A^* \) be \( B \), \( \neg(B \lor C) \), \( \neg(\forall x \leq t)B \) or \( \neg(\exists x \leq t)B \) respectively.
Then

\[ \text{WITNESS}_A^F(w, \bar{x}) = \text{WITNESS}_A^F(w, \bar{x}). \]

Definition: Let \( A(\bar{x}) \) be a \( \Sigma_1^1(PV) \)-formula. We say that \( PV \) essentially proves \( (\forall \bar{x})A(\bar{x}) \) if there is a \( PV \)-function symbol \( F \) such that

\[ PV \vdash \text{WITNESS}_A^F(F(\bar{x}), \bar{x}) \neq 0. \]

We shall see below that if \( A \in \Sigma_1^1(PV) \) and \( S_A^F(PV) \vdash (\forall \bar{x})A(\bar{x}) \) then \( PV \) essentially proves \( (\forall \bar{x})A(\bar{x}) \).

Proposition 5: Let \( A(t, \bar{x}) \) be a \( \Sigma_1^1(PV) \)-formula and let \( B(\bar{x}) \) be \( A(t(\bar{x}), \bar{x}) \) for some term \( t \). Then \( PV \) proves

\[ \text{WITNESS}_A^F(w, \bar{x}) \neq 0 \supset \text{WITNESS}_A^F(w, t(\bar{x}), \bar{x}) \neq 0. \]

Proof: by induction on the complexity of \( A \). \( \Box \)

Proposition 6: Let \( A \) be a \( \Sigma_1^1(PV) \)-formula and let \( \bar{A} \) be a vector of free variables containing all the free variables of \( A \). Then there are functions \( \text{MINWIT}_A^F \) and \( \text{WITSIZE}_A^F \) definable by \( PV \) such that

\[ PV \vdash \text{MINWIT}_A^F(v) \leq \text{WITSIZE}_A^F(\bar{x}) \]

and

\[ PV \vdash \text{WITNESS}_A^F(w, \bar{x}) \neq 0 \supset \text{WITNESS}_A^F(\text{MINWIT}_A^F(w), \bar{x}) \neq 0. \]

So \( \text{MINWIT}_A^F \) maps any witness for \( A(\bar{A}) \) to a minimal witness; the Gödel number of the minimal witness is bounded uniformly by \( \text{WITSIZE}_A^F(\bar{A}) \).

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 \( \langle n_1, \ldots, n_m \rangle \) 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 \( \beta \) function, for any reasonable definition of the \( \beta \) 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 §5.5 apply here; in particular, $S_3(PV)$ is a natural deduction theory.

**Theorem 7:** Suppose $S_3(PV)$ proves the sequent $\Gamma, \Pi \rightarrow \Lambda, \Delta$ and that each formula in $\Gamma \cup \Delta$ is a $\Sigma^0_1(PV)$-formula and each formula in $\Pi \cup \Lambda$ is a $\Pi^0_1(PV)$-formula. Let $c_0, \ldots, c_n$ be the free variables in $\Gamma, \Pi \rightarrow \Lambda, \Delta$. Let $X$ and $Y$ be the formulae

\[
X = (\exists \Delta') \exists A \neg C : C \in A
\]

\[
Y = (\forall \Delta') \exists C : C \in A
\]

Then there is a $PV$-function symbol $F$ such that

\[
PV \vdash \text{WITNESS}^{\bar{w}}_X(w, \bar{x}) \neq 0 \supset \text{WITNESS}^{\bar{w}}_Y(F(w, \bar{x}), \bar{c}) \neq 0.
\]

It is immediately obvious that Theorem 4 follows from Theorem 7, since when $A$ is atomic,

\[
PV \vdash \text{WITNESS}^{\bar{w}}_A(w, \bar{x}) \neq 0 \supset A(\bar{x}).
\]

**Proof of Theorem 7.**

By Proposition 4.8, there is a free cut free $S_3(PV)$-proof $P$ of $\Gamma, \Pi \rightarrow \Lambda, \Delta$ which is in free variable normal form. Every formula of $P$ is in $\Sigma^0_1(PV) \cup \Pi^0_1(PV)$ and each cut formula of $P$ is a $\Sigma^0_1(PV)$-formula. As in the proof of Theorem 5.5, we assume without loss of generality that $\Pi$ and $\Lambda$ 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 \textit{Basic} axiom, or (c) one of the axioms of $S_3(PV)$ 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 (5): (\because:\text{left}). Suppose the last inference of \( P \) is:

\[
\frac{B, \Gamma^* \rightarrow \Delta}{B \land C, \Gamma^* \rightarrow \Delta}
\]

Let \( D \) be the formula \( B \land (\overline{\Gamma}^*) \) and let \( E \) be \( (B \land C) \land (\overline{\Theta}^*) \). The induction hypothesis is that for some \( PV \)-function symbol \( G \),

\[
PV \vdash \text{WITNESS}_{\overline{\Theta}^*}(G(w, \overline{\Theta}), \overline{\Theta}) \neq 0.
\]

Let \( H \) be the \( PV \)-function symbol defined so that \( H(w) = \langle \beta(1, \beta(1, w)), \beta(2, w) \rangle \) and let \( F(w, \overline{\Theta}) = G(H(w), \overline{\Theta}) \). Then

\[
\begin{align*}
PV \vdash & \text{WITNESS}_{\overline{\Theta}^*}(G(w, \overline{\Theta}), \overline{\Theta}) \neq 0, \\
PV \vdash & \text{WITNESS}_{\overline{\Theta}^*}(F(w, \overline{\Theta}), \overline{\Theta}) \neq 0,
\end{align*}
\]

which is what we wanted to show.

Cases (6)-(7): Omitted.

Case (6): (\because:\text{right}). Suppose the last inference of \( P \) is:

\[
\frac{\Gamma \rightarrow B, \Delta^*}{\Gamma \rightarrow B \land C, \Delta^*}
\]

Let \( D \) be the formula \( B \land (\overline{\Delta}^*) \), let \( E \) be \( C \land (\overline{\Delta}^*) \) and let \( F \) be \( (B \land C) \land (\overline{\Delta}^*) \). The induction hypothesis is that there are \( PV \)-functions \( G \) and \( H \) such that

\[
\begin{align*}
PV \vdash & \text{WITNESS}_{\overline{\Theta}^*}(G(w, \overline{\Theta}), \overline{\Theta}) \neq 0, \\
PV \vdash & \text{WITNESS}_{\overline{\Theta}^*}(H(w, \overline{\Theta}), \overline{\Theta}) \neq 0.
\end{align*}
\]

Define the \( PV \)-function \( K \) so that

\[
K(v, w, \overline{\Theta}) = \begin{cases} 
\nu & \text{if } \text{WITNESS}_{\overline{\Theta}^*}(v, \overline{\Theta}) \neq 0 \\
\nu & \text{otherwise}
\end{cases}
\]
and $F$ so that

$$F(w,\overline{x}) = \langle \beta(1, G(w, \overline{x})), \beta(1, H(w, \overline{x})), K(\beta(2, G(w, \overline{x})), \beta(2, H(w, \overline{x})), \overline{x}) \rangle.$$ 

It is easy to see that

$$PV \vdash \text{WITNESS}_{\overline{\alpha}}^\mathcal{A}(w, \overline{x}) \neq 0 \supset \text{WITNESS}_{\overline{\alpha}}^G(F(w, \overline{x}), \overline{x}) \neq 0.$$ 

Case (9): $(\exists \overline{x}; \text{right})$. Omitted.

Case (10): $(\forall \overline{x}; \text{right})$. Suppose the last inference of $P$ is:

$$a \leq \{r\}, \Gamma \vdash B(a) \Delta^* \quad \Gamma \vdash (\forall x \leq \{r\}) B(x) \Delta^*$$

where $a$ is the eigenvariable and must not appear in the lower sequent. Let $D$ be the formula $a \leq \{r\}(\Delta^*)$, let $E$ be $B(a) (\forall x \leq \{r\}) B(x) (\forall x \leq \{r\}) B(x)$, and let $C$ be $V \leq \{r\} B(x)$ (or $\forall \Delta^*$). By the induction hypothesis there is a $PV$-function $G$ so that

$$PV \vdash \text{WITNESS}_{\overline{\alpha}}^D(w, \overline{x}, a) \neq 0 \supset \text{WITNESS}_{\overline{\alpha}}^G(G(w, \overline{x}, a), \overline{x}) \neq 0.$$ 

By Proposition 1, there is a $PV$-function $H$ such that $PV$ proves

$$\beta(0, H(w, \overline{x})) = \{r\} + 1: 0 \neq \{r\}(\beta(0, H(w, \overline{x})) = \beta(1, G(w, \overline{x}, x))).$$

We define the $PV$-function symbol $J$ as follows by limited iteration on notation:

$$J(w, \overline{x}, 0) = \beta(2, G(w, \overline{x}, 0))$$

$$J(w, \overline{x}, i(z)) = \begin{cases} J(w, \overline{x}, x) & \text{if} \quad \text{WITNESS}_{\overline{\alpha}}^J(J(w, \overline{x}, x), \overline{x}) \neq 0 \\ \beta(2, G(w, \overline{x}, x), \overline{x}) & \text{otherwise} \end{cases}$$

Then define $F(w, \overline{x}) = \langle H(w, \overline{x}), J(w, \overline{x}, r(\overline{x})), \overline{x} \rangle$. $PV$ can use an induction on notation argument to prove

$$\text{WITNESS}_{\overline{\alpha}}^\mathcal{A}(w, \overline{x}) \neq 0 \supset \text{WITNESS}_{\overline{\alpha}}^G(F(w, \overline{x}), \overline{x}) \neq 0.$$

Case (12): ($\Sigma^I_1$-PND). Suppose the last inference of $P$ is

$$B([\lambda a]t),\Gamma^* \Rightarrow B(a),\Delta^*$$

where the eigenvariable $a$ must not appear in the lower sequent. We only consider the case where $B(0)$ is in $\Gamma$ and $B(1)$ is in $\Delta$.

Let $D$ be the formula $B([\lambda a]t)(\overline{A}^\Delta^*)$, let $E$ be $B(a)(\overline{A}^\Delta^*)$, let $C$ be $B(0)(\overline{A}^\Delta^*)$ and let $A$ be $B(1)(\overline{A}^\Delta^*)$. By the induction hypothesis, there is a $PV$-function $G$ such that

$$PV \vdash WITNESS_{\overline{A}^\Delta^*}(w,\overline{A}^\Delta^*,a) \neq 0 \supset WITNESS_{\overline{A}^\Delta^*}(G(w,\overline{A}^\Delta^*,a),a) \neq 0.$$ 

Let $TRM$ be the $PV$-function satisfying $TRM(x,i)\equiv MSP(x,i)$. Let $J(v,w) \equiv \langle J(1,w),J(2,v) \rangle$. Let $H$ be the $PV$-function defined by the following limited iteration on notation:

$$H(w,\overline{A},0) = MINWIT_{\overline{A}}(G(w,\overline{A},0))$$

$$H(w,\overline{A},s_{1}(x)) =
\begin{cases} 
H(w,\overline{A},x) & \text{if } WITNESS_{\overline{A}}(J(2,H(w,\overline{A},x)),\overline{A}^\Delta^*) \neq 0 \\
& \text{or if } |x| \geq |\overline{A}^\Delta^*| \\
MINWIT_{\overline{A}}(G(J(w,H(w,\overline{A},x)),\overline{A}^\Delta^*),TRM(t(\overline{A}),s_{1}(x),i)) & \text{otherwise}
\end{cases}$$

This is a valid limited iteration on notation definition since

$$PV \vdash H(w,\overline{A},x) \leq WITNESS_{\overline{A}}(\overline{A}^\Delta^*,a)$$

because $MCN\overline{A}^{\Delta^*}$ was used in the definition of $H$. By using induction on notation, $PV$ can prove

$$[x]_{A} \leq [t(\overline{A})] \land WITNESS_{\overline{A}}(w,\overline{A}^\Delta^*) \neq 0 \supset WITNESS_{\overline{A}}(H(w,\overline{A},x),\overline{A}^\Delta^*),TRM(t(\overline{A}),s_{1}(x),i)) \neq 0.$$ 

So define $F(w,\overline{A}) \equiv H(w,\overline{A},s_{1}(t(\overline{A})))$ and then

$$PV \vdash WITNESS_{\overline{A}}(w,\overline{A}^\Delta^*) \neq 0 \supset WITNESS_{\overline{A}}(F(w,\overline{A}),\overline{A}^\Delta^*) \neq 0.$$
Q.E.D. □

**Corollary 8:** Let $A(\bar{x})$ be a $\Sigma^0_1(PV)$-formula. If $S^1_2(PV) \vdash (\forall \bar{y})A(\bar{x})$ then $PV$ essentially proves $(\forall \bar{x})A(\bar{x})$.

**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 $S_1$. 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 $S_1$. 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, $S_1$ can $\Sigma_1$-define any polynomial time algorithm. This might appear to be an a priori argument that the arithmetization of metamathematics can be carried out in $S_1$. 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 $S_1$ and using this framework to outline how the arithmetization of metamathematics in $S_1$ can be carried out intensionally.

7.1. Trees.

As a preliminary we need to give a method for coding trees in $S_1$. 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 $\Sigma_1$-definable functions and $\Delta_1$-predicates of $S_1$:

(a) $RBracket = 0$
$LBracket = 1$
$Node(x) \iff x \geq 2$

(b) $Balanced(w) \iff$\[
\begin{align*}
(\#j < \text{Len}(w)) & (LBracket=\text{R}(S_j,w)) = (\#j < \text{Len}(w)) (RBracket=\text{R}(S_j,w)) \\
& \land (\forall i < \text{Len}(w)) (\#j \leq i (RBracket=\text{R}(S_j,w)) \leq (\#j \leq i (LBracket=\text{R}(S_j,w)))
\end{align*}
\]

Note that the counting operations are all equivalent to length bounded counting and hence by Theorem 2.7, $Balanced$ is a $\Delta_1$-definable predicate. We shall use length bounded counting freely without comment from now on.

116
\[\begin{array}{c}
\begin{array}{c}
\text{a} \\
\text{c} \\
\text{f} \\
\end{array}
\begin{array}{c}
\text{b} \\
\text{d} \\
\text{e} \\
\text{g} \\
\text{i} \\
\text{j} \\
\text{k}
\end{array}
\end{array}\]

A tree is coded by a sequence which enumerates the tree in depth first order. Two special symbols, \(\wedge\) and \(\vee\) are used to denote movement down and up the tree. The tree shown has two roots, labeled a and b.

Figure 2

(c) \(\text{Depth}(i, w) = (\#j < i)(\text{LBracket} = \text{S}(S_j, w)) - (\#j < i)(\text{RBracket} = \text{S}(S_j, w))\)

(d) \(\text{MultiTree}(w) \iff \text{Seq}(w) \wedge \text{Len}(w) \neq 0 \wedge \text{Balanced}(w) \wedge \text{LBracket} \neq \text{S}(1, w) \wedge \forall i < \text{Len}(w) \exists \text{LBracket} = \text{S}(i+1, w) \wedge \text{Node}(\text{S}(i+2, w))\)

\(\text{MultiTree}(w)\) is true iff \(w\) codes a tree, which may have more than one root.

(e) \(\text{Tree}(w) \iff \text{MultiTree}(w) \wedge (\exists i < \text{Len}(w))(i > 0 \wedge \text{Node}(\text{S}(S_j, w)) \wedge \text{Depth}(S_j, w) = 0)\)

(f) \(\text{Leaf}(i, w) \iff \text{MultiTree}(w) \wedge \text{Node}(\text{S}(i, w)) \wedge \text{LBracket} = \text{S}(S_i, w)\)

So \(\text{Leaf}(i, w)\) is true iff \(S(i, w)\) 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 \(\text{Father} \) and \(\text{Son}\) so that if \(\text{Father}(i, w) = j\) then the node \(\text{S}(i, w)\) is the father of the node \(\text{S}(j, w)\), and so that \(\text{Son}(j, w) = i\) iff \(\text{S}(i, w)\) is the \(n\)-th son of the node \(\text{S}(j, w)\).

(g) \(\text{Father}(i, w) =\)

\[\begin{cases}
0 & \text{if } \text{Node}(i, w)\text{ balanced (Subseq}(w, j+2, i)) \text{ and } \text{Len}(w) + 1 \\
0 & \text{otherwise}
\end{cases}\]

We use \(\text{Len}(w) + 1\) as the alternative value for the function \(\text{Father}\) since \(\text{Len}(w) + 1\) is never a node and hence never a valid father.

(h) \(\text{SonPos}(i, j, w) =\)

\[\begin{cases}
0 & \text{if } \text{Father}(i, w) = j \\
0 & \text{otherwise}
\end{cases}\]
\[ \text{Son}(k,j,w) = (\mu i \leq \text{Len}(w)) \text{[SonPar}(i,j,w) \rightarrow k) \]

Note that the father of a root of a multtree is 0 and that the roots of a multtree are the sons of an imaginary node as the zeroth position of the sequence coding the multtree.

(i) \[ \text{Valence}(j,w) = (\# x \leq \text{Len}(w)) (\text{Father}(z,w) = j) \]

(ii) \[ \text{Son}(k,j,w) = (\beta(\text{Son}(k,j,w),w) = 2) \]

\[ \text{Father}(i,w) = \beta(\text{Father}(i,w),w) = 2 \]

\[ \text{Root}(w) = \beta(1,w) = 2 \]

\[ \text{Node}(i,w) = \beta(i,w) = 2 \]

We subtract 2 so that the values of the node labels are distinct from the codes for brackets, namely 6 and 1 for "[" and "]".

(k) We also define a function for extracting subtrees of trees:

\[ \text{SubTree}(i,w) = \text{Subseq}(i, \max(j \leq \text{Len}(w)+1 : \text{Tree}(\text{SubSeq}(i,j,w))),w) \].

The above encoding of trees is intensional in the sense of Feferman [9]. The skeptical reader may verify that, for instance, \( S^2 \) can prove

\[ \text{MultiTree}(w) \land \text{Node}(\beta(j,w)) \supset (\text{Leaf}(j,w) \equiv \text{Valence}(j,w) = 0) \]

\[ \text{MultiTree}(w) \land \text{Node}(\beta(i,w)) \supset (\text{Depth}(\text{Father}(i,w),w) = \text{Depth}(i,w) - 1) \]

\[ \text{MultiTree}(w) \land \text{Node}(i,w) \supset \text{Tree}(\text{SubTree}(i,w)) \].

7.2. Inductive Definitions.

We show in this section that \( S^2 \) 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 \( S^2 \) to be carried out by induction on the complexity of an inductive definition. We later use the constructions of this section to argue that \( S^2 \) can arithmetize metamathematics.
Definition: The \( n \) predicates \( P_0, \ldots, P_{n-1} \) are defined by a \( p \)-inductive definition if 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 \( Q_s \):

\[
Q_s(x) \iff R_{s0}(f_{s0}(x)) \land \cdots \land R_{sn}(f_{sn}(x))
\]

where the following conditions hold:

(i) each \( R_{sj} \) is \( P_i \) or \( \neg P_i \) for some \( i < n \),
(ii) each \( f_{sj} \) is a \( \Sigma^{0}_1 \)-definable function of \( S_{i}^{1} \),
(iii) for each \( j \leq i \), \( S_{i}^{1} + \neg \forall x \exists y \phi(x,y) \leq \neg \forall x \exists y \phi(x,y) \leq k \).
(iv) \( S_{i}^{1} + \forall x (\exists y \phi(x,y)) \leq k \).
(c) For each \( i < n \) there is a function \( g_i \) which is \( \Sigma^{0}_1 \)-definable in \( S_{i}^{1} \) such that \( S_{i}^{1} + \exists y (\exists x g_i(x,y)) \leq k \).
(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

\[
P_i(x) \iff Q_{i,i}(x).
\]

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. This algorithm can be formalized in \( S_{i}^{1} \) 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 \( P_i \) in \( \Delta^{0}_1 \)-definable in \( S_{i}^{1} \).

Proof: In order to \( \Delta^{0}_1 \)-define the predicates \( P_i \) in \( S_{i}^{1} \) 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 nodes of such a node must provide evidence that \( P_m(y) \) or \( \neg P_m(y) \) respectively is valid. For example, if \( g(x)=s \), then the nodes of \( < P_m,y > \) must be by \( i \), nodes labeled \( R_{sj}f_{sj}(y) \) for \( j \leq i \). 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 \( < P_i,1 > \) or \( \neg < P_i,1 > \).

We begin by writing out a formal definition for a "demonstration tree" for \( P_i(x) \). Let \( C_{i,j}: \overline{C}_{i,j}, B_{i,j} \) and \( D_i \) be fixed terms defined by:

\[
C_{i,j} = \begin{cases} \ I_i & \text{if } R_{i,j} \in P_i, \\ I_{i+n} & \text{if } R_{i,j} \in \neg P_i \end{cases}
\]


Gödel incompleteness Theorems

\[ \forall_{x,y} \left\{ \begin{array}{l} I_i \\
I_{i+n} \end{array} \right. \quad \text{if } R_{x,y} \text{ is } P_i \\
I_{i+n} \quad \text{if } R_{x,y} \text{ is } P_i \right. \\
B_{x,y} = I_i \quad \text{where } R_{x,y} \in \neg P_i \text{ or } P_i \\
B_{x,y} = I_i \quad \text{if } P(0) \\
B_{x,y} = I_{i+n} \quad \text{if } \neg P(0) \right. \\

(Recall that \( I_i \) is a term with value \( j \).) The leaves of a "demonstration tree" must satisfy the leaf condition:

\[ DTLC(u,w) = (\forall_{i \leq n} \beta(1, \text{Node}(u,w)) = D_i) \land \beta(2, \text{Node}(u,w)) = 0 \]

or, in words, a leaf must be labeled \( \langle D_i, 0 \rangle \) for some \( i \). The non-leaf nodes of the "demonstration tree" must satisfy the following condition:

\[ DTNC(u,w) \iff \beta(1, \text{Node}(u,w)) < 2n \land \beta(2, \text{Node}(u,w)) \neq 0 \]

\[ \land \bigwedge_{i=0}^{n-1} \left( \text{Valence}(u,w) = i \iff \bigwedge_{i=0}^{n} \bigwedge_{j=0}^{m} \beta(1, \text{Node}(u,w)) \bigwedge_{i=0}^{n} \beta(2, \text{Son}(S_j, u, w)) \bigwedge_{i=0}^{n} \beta(3, \text{Son}(S_j, u, w)) \right) \]

We combine both these requirements in

\[ DTNC(u,w) \iff (\text{Leaf}(u,w) \land DTLC(u,w)) \lor (\neg \text{Leaf}(u,w) \land DTNC(u,w)) \]

So a "demonstration tree" which proves \( P(x) \) or \( \neg P(x) \) must satisfy

\[ \text{DemoTree}(u,w) \iff (\exists w) \left( (u < \text{Len}(w)) \land (\text{Node}(S_u, w) \land DTNC(S_u, w)) \right) \]

\[ \land \left( \text{Root}(w) = i \iff i > \forall \text{Root}(w) = i + n \right) \]

We will introduce \( P_i \) in \( S_1 \) as a \( \Delta_1 \)-defined predicate symbol by:

\[ P_i(x) \iff (\exists w) \left( \text{DemoTree}(w,x) \land \beta(1, \text{Root}(w)) = i \right) \]

\[ \iff (\exists w) \left( \text{DemoTree}(w,x) \land \beta(1, \text{Root}(w)) = i + n \right) \]
Thus it will suffice to establish that $S^2_2$ proves

$$(\forall x)(\exists w)\text{DemoTree}_t(w,x)$$

and

$$\text{DemoTree}_t(w,x) \supset (1,\text{Root()}(w)) \supset (1,\text{Root()}(v)).$$

Since it is easier, we first show that $S^2_2$ proves the uniqueness condition. We argue informally inside the theory $S^2_2$. Suppose $w$ and $v$ are $\text{DemoTree}_t$'s for $P(x)$ and/or $\neg P(x)$. Let $A(w,v,b)$ be the formula

$$(\forall u < b)(\neg\text{Node}(Su,w) \supset \text{Beta}(Su,w) \supset \text{Beta}(Su,v)) \land$$

$$\text{Node}(Su,w) \supset \text{Beta}(2,\text{Node}(Su,v)) \lor \text{Beta}(2,\text{Node}(Su,v)).$$

It follows from the definition of $\text{DemoTree}_t$ that $A(w,v,b) \supset A(w,v,b)$. Hence, by $\Sigma^L_2$-LIND, $\beta(1,\text{Node}(SU(w),v))$ and $\beta(1,\text{Node}()(SU(w)))$ and hence $\text{Len}(v) = \text{Len}(w)$. Now let $B(w,v,b)$ be the formula

$$(\forall u < \text{Len}(w))(u \geq \text{Len}(w) - b \land \text{Node}(Su,w) \supset \text{Beta}(1,\text{Node}(SU(w),v)) \supset \text{Beta}(1,\text{Node}(SU(w))).$$

Now it follows that $B(w,v,b) \supset B(w,v,b)$ so by $\Sigma^L_2$-LIND, $\beta(1,\text{Node}(SU(w),v))$. But this immediately tells us that

$$\beta(1,\text{Root()}(w)) \supset \beta(1,\text{Root()}(v)).$$

This completes the $S^2_2$-proof of the uniqueness condition.

The rest of the proof of Theorem 1 is devoted to establishing the existence condition for $\text{DemoTree}_t$. It is tempting to just argue by induction on the length of $x$ that a $\text{DemoTree}_t$ exists. Unfortunately, this argument would use $\Pi^L_2$-IND and we can not carry this out in $S^2_2$. Instead we must use a more sophisticated argument to construct the $\text{DemoTree}_t$. What we will do is formalize a breadth first algorithm which constructs the demonstration tree and then labels the nodes properly. We first define:

$$\text{PDTNC}(u,w) \iff \beta(1,\text{Node}()(u,w)) < n \land \beta(2,\text{Node}()(u,w)) \neq 0.$$

$$\land \bigwedge_{s=0}^{n-1} \bigwedge_{j=0}^{t_i} \beta(1,\text{Node}()(u,w)) \land \beta(2,\text{Node}()(u,w)) = s \supset$$

$$\supset \text{Valence}(u,w) = i + 1 \land \bigwedge_{j=0}^{t_i} \beta(2,\text{Node}()(u,w)) \land$$

$$\bigwedge_{j=0}^{t_i} \beta(1,\text{Son}()(Sj,u,w)) \land \beta(1,\text{Son}()(Sj,u,w)))$$

$$\land \bigwedge_{j=0}^{t_i} \beta(1,\text{Son}()(Sj,u,w))$$

$$\land \text{PDTLC}(u,w) \iff \beta(1,\text{Node}()(u,w)) < n \land \beta(2,\text{Node}()(u,w)) = 0.$$
Gödel Incompleteness Theorems

\[ PTDTCI(u,w) \iff (\text{Leaf}(u,w) \supset PTDLCI(u,w)) \land (\neg \text{Leaf}(u,w) \supset PTDNCI(u,w)) \]

\[ PTDI(u,x,b) \iff \text{Tree}(w) \land \text{Root}(w) - i, r > ^0 \]

\[ \forall u < \text{Len}(w) \forall \text{Node}(u,w) \forall \text{Depth}(u,w) = k \geq PTDTCI(w,u) \]

\[ \forall u < \text{Len}(w) \forall \text{Node}(u,w) \forall \text{Depth}(u,w) = k \geq PTDNCI(w,u) \]

\[ \forall \text{SizeBounds} \]

where we explain \text{SizeBounds} below. So \( PTDI(w,x,b) \) asserts that \( w \) is a tree containing the first \( b+1 \) levels of the construction of a demonstration tree. The \text{SizeBounds} is a formula which bounds the size of \( w \). What we wish to show is

\[ S^2 \{ \{ w \leq t(x,b) \} \mid \text{PTDI}(w,x,b) \} \subset \{ w \leq t(x,b) \} \text{PTDI}(w,x,b) \]

for some term \( t \); the \text{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 \( y > 0 \). The number of bits used to code such a node can be required to be not more than \( 4 \cdot |x| + 4 \cdot |y| + 4 \cdot |i| + 2 \cdot |x| \cdot |y| + 4 \cdot |y| + 10 \). We add on an adjustment allowing for the bits needed to code two brackets and conclude that each node can be coded by \( 4 \cdot |x| + 18 + 4 \cdot |y| \). 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

\[
\sum_{\text{Depth}(<i,y>) = c} (4|y|+4|n|+18) \leq 4 \sum_{|y|} |y| + |x| (4|n|+18) \\
\leq |x| (4|n|+18) \leq |x| (4|n|+22).
\]

Since the tree has depth \( b \), the total number of bits required to code the non-leaf nodes of the tree is \( \leq (b+1) |x| (4|n|+22) \).

We must also consider the nodes labeled \( <i,0> \). Let \( i_{\text{max}} \) be \( \max\{i_x : x = 0, \ldots, k\} \). There are \( \leq b \cdot |x| \) non-leaf nodes on the first \( b \) levels of the tree and below them are \( \leq b \cdot |x| (i_{\text{max}} + 1) \) leaf nodes. (The extra 1 is for the case \( x = 0 \).) Since a label \( <i,0> \) and its surrounding brackets can be coded by \( 4 \cdot |n|+18 \), the total number of bits used to code these nodes is bounded by

\[
(1 + b \cdot |x| (1 + i_{\text{max}})) (4|n|+18).
\]

So the length \( |w| \) of \( w \) is bounded by

\[
s(x,b) = (1 + |x| (1 + b (2 + i_{\text{max}}))) (4|n|+22).
\]

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 \( \text{SizeBounds} \) 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 \( \text{SizeBounds} \), so we leave it as an exercise for the skeptical reader. Given that \( \text{SizeBounds} \) is properly formulated, it is now straightforward for \( S_2^I \) to prove

\[
(\exists w \leq |(x,b)|) \text{PDT}_I(w,x,b) \supset (\exists x \leq |(x,b)|) \text{PDT}_I(w,x,b).
\]

So by \( \Sigma^I_1\text{-LIND} \), \( S_2^I \vdash (\exists w \leq |(x,b)|) \text{PDT}_I(w,x,b) \). Finally, we need to show that

\[
S_2^I \vdash (\exists w) \text{PDT}_I(w,x,x) \supset (\exists w) \text{DemoTree}_I(w,x).
\]

So let \( C(w,v,b) \) be the formula

\[
\begin{align*}
\text{Len}(w) & = \text{Len}(w) \land (\forall u \leq \text{Len}(w)) \left( u < \text{Len}(w) \supset \beta(S_u,u) = \beta(S_u,v) \right) \\
\beta(u \geq \text{Len}(w) \supset \beta, \text{Node}(S_u,u) = \beta(S_v,u) \supset \beta(S_v,v)) & \land \\
\beta(u \geq \text{Len}(w) \supset \beta, \text{Node}(S_u,u) = \text{DTNC1}(S_u,v) \supset \\
\beta(2, \text{Node}(S_v,u) = \beta(S_v,v)) & \land \\
\beta(1, \text{Node}(S_v,u) = \text{Rem}(1, \text{Node}(S_v,v), n)) & \land
\end{align*}
\]

It is quite easy to see that \( S_2^I \vdash \text{PDT}_I(w,x,x) \supset (\exists w \leq w) C(w,v,0) \) and

\[
S_2^I \vdash \text{PDT}_I(w,x,x) \land (\exists w \leq 2^{2^{\text{Len}(w)}}) C(w,v,b) \supset (\exists x \leq 2^{2^{\text{Len}(w)}}) C(w,v,b+1).
\]

Hence, by \( \Sigma^I_1\text{-LII}/D \),

\[
S_2^I \vdash \text{PDT}_I(w,x,x) \supset (\exists w \leq 2^{2^{\text{Len}(w)}}) C(w,v,\text{Len}(w))
\]

from which \( S_2^I \vdash (\exists w) \text{DemoTree}_I(w,x) \) is immediate.

Q.E.D. \( \Box \)

Theorem 1 states that \( S_2^I \) can \( \Delta^I_1 \)-define predicates which have p-inductive definitions. We also want \( S_2^I \) to be able to prove theorems involving \( \Delta^I_1 \)-defined predicates. Accordingly we need to know that certain kinds of inductive proofs can be formalized in \( S_2^I \).

**Definition:** Let \( P_0, \ldots, P_{\beta+1} \) be defined p-inductively as above. We say that the 2\( \beta \) formulae
Gödel Incompleteness Theorems

\(( \forall x)(P(x) \supset B(x)) \)  
\(( \forall x)(\neg P(x) \supset C(x)) \)

have a p-inductive proof if the following hold:

(a) Each \( B_i \) and \( C_i \) is \( \Delta^1_1 \) with respect to \( S^1_2 \).

(b) For \( 0 \leq i \leq k \), \( 0 \leq j \leq i' \), \( P_{i,j} \) is as in clause (b) of the p-inductive definition for \( P_0, \ldots, P_{i,j} \). Let \( Q_{i,j} \) be the formula \( P_i \equiv R_{i,j} = P_i \) or the formula \( C_i \) if \( R_{i,j} = \neg P_i \). Define \( Q_{i,j} \) dually to be \( B_i \) if \( R_{i,j} = \neg P_i \) and so be \( C_i \) if \( R_{i,j} = P_i \).

(c) For \( i = 0, \ldots, n - 1 \), and \( 0 \leq i \leq k \), \( S^1_2 \) proves

\[ s(x) = e \land \prod_{i=0}^{i+1} (R_{i,j}(f_{i}(x)) \land Q_{i,j}(f_{i}(x)))) \supset B_i(x) \]
\[ s(x) = e \land \prod_{i=0}^{i+1} (\neg R_{i,j}(f_{i}(x)) \land Q_{i,j}(f_{i}(x)))) \supset C_i(x). \]

**Theorem 2:** Given (a), (b), (c) as above, \( S^1_2 \) proves

\(( \forall x)(P_i(x) \supset B_i(x)) \land (\neg P_i(x) \supset C_i(x)) \)

for \( 0 \leq i < n \).

**Proof:** Let \( A(w,a,b) \) be the formula

\[
\text{DemoTree}(w,a) \supset (\forall u < \text{Len}(w))(u \geq \text{Len}(w) \land \text{Node}(Su,w) \supset \\
\prod_{i=0}^{n-1} [\beta(1,\text{Node}(Su,w)) \equiv \neg B_i(\beta(2,\text{Node}(Su,w)))] \land \\
\prod_{i=0}^{n-1} [\beta(1,\text{Node}(Su,w)) \equiv \neg i \lor C_i(\beta(2,\text{Node}(Su,w)))).
\]

Clearly, \( S^1_2 \vdash A(w,a,0) \). Also, because of clause (c) of the p-inductive proof, \( S^1_2 \) proves \( A(w,a,b) \supset A(w,a,b') \). Hence, by \( \Sigma^1_1 \text{-LIND} \), \( S^1_2 \vdash A(w,a,\text{Len}(w)) \). By Theorem 1, \( S^1_2 \vdash \text{DemoTree}(w,a) \) 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. \( \Box \)
**Definition:** The function \( F \) is defined by a \( p \)-**inductive definition** if \( F \) is defined by the following:

1. \( k \) is a fixed nonnegative integer.
2. For each \( s \leq k \) there is an \( i_s \), any function \( G_s \), and \( i_s \) unary functions \( f_s, \ldots, f_{s,i_s} \) satisfying:
   1. \( G_s \) and each \( f_s \) are \( \Sigma^1_2 \)-definable functions of \( S^1_2 \).
   2. \( S^1_2 \vdash \forall x \exists y [F(x,y) \equiv y] \) for all \( j \leq i_s \).
   3. \( S^1_2 \vdash \forall x [f(x,z) \leq k] \).
3. There is a function \( g \) \( \Sigma^1_2 \)-definable by \( S^1_2 \) so that \( S^1_2 \vdash (\forall x) (g(x) \leq k) \).
4. \( F(x) \) is defined inductively by:
   \[
   F(x) = G_s(\bar{F}(G(x),z), \ldots, F(u)(s,z)).
   \]
5. There is a term \( t(x) \) (which will bound \( F(x) \)) so that for all \( s \leq k \),
   \[
   S^1_2 \vdash \forall x \left( \forall y \left( t(x) \leq G(x) \right) \implies \forall y \left( G(x) \leq y \right) \right).
   \]

**Theorem 3:** Let \( F \) be defined by the \( p \)-inductive definition above. Then \( F \) is \( \Sigma^1_2 \)-definable in \( S^1_2 \). Furthermore, the definition of \( F \) in \( S^1_2 \) is intensionally correct in that properties of \( F \) can be proved in \( S^1_2 \) 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. \( \square \)

### 7.3. The Arithmetization of Metamathematics.

In order to establish the Gödel incompleteness theorems for Bounded Arithmetic, we need to introduce \( \Delta^1_2 \)-defined function symbols and \( \Delta^1_2 \)-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**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>∀</code></td>
<td>0</td>
</tr>
<tr>
<td><code>∃</code></td>
<td>1</td>
</tr>
<tr>
<td><code>¬</code></td>
<td>2</td>
</tr>
<tr>
<td><code>∧</code></td>
<td>3</td>
</tr>
<tr>
<td><code>→</code></td>
<td>4</td>
</tr>
<tr>
<td><code>∀</code></td>
<td>5</td>
</tr>
<tr>
<td><code>∃</code></td>
<td>6</td>
</tr>
<tr>
<td><code>¬</code></td>
<td>7</td>
</tr>
<tr>
<td><code>∧</code></td>
<td>8</td>
</tr>
<tr>
<td><code>→</code></td>
<td>9</td>
</tr>
</tbody>
</table>

**Non-logical symbols**

<table>
<thead>
<tr>
<th>Category</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Constants</td>
<td>0-18</td>
</tr>
<tr>
<td>Unary Functions</td>
<td>19-32</td>
</tr>
<tr>
<td>Binary Functions</td>
<td>19-32</td>
</tr>
<tr>
<td>Binary Relations</td>
<td>1-18</td>
</tr>
</tbody>
</table>

**Free Variables**

- `a_1`: 19
- `a_2`: 23
- `a_3`: 27
- `x_1`: 17
- `x_2`: 21
- `x_3`: 25
- `x_4`: 29
- `y_1`: 32
- `y_2`: 36
- `z_1`: 40

**Bound Variables**

- `x_0`: 40
- `y_0`: 40
- `z_0`: 40
- `x_1`: 40
- `y_1`: 40
- `z_1`: 40
- `x_2`: 40
- `y_2`: 40
- `z_2`: 40

Corresponding to this assignment of Gödel numbers we introduce the following predicate symbols in $S^2$:

- $AxQuant(x) \iff x=0$
- $ExQuant(x) \iff x=1$
- $Nof(x) \iff x=2$
- $Imp(x) \iff x=3$
- $And(x) \iff x=4$
- $Or(x) \iff x=5$
- $Logic(x) \iff x=6$
- $Div(x) \iff x=7$
- $Times(x) \iff x=8$
- $Plus(x) \iff x=9$
- $Succ(x) \iff x=10$
- $Logic(x) \iff x=11$
- $Div(x) \iff x=12$
- $Times(x) \iff x=13$
- $Plus(x) \iff x=14$
- $Succ(x) \iff x=15$
- $Logic(x) \iff x=16$
- $Div(x) \iff x=17$
- $Times(x) \iff x=18$
- $Plus(x) \iff x=19$
- $Succ(x) \iff x=20$
- $Logic(x) \iff x=21$
- $Div(x) \iff x=22$
- $Times(x) \iff x=23$
- $Plus(x) \iff x=24$
- $Succ(x) \iff x=25$
- $Logic(x) \iff x=26$
- $Div(x) \iff x=27$
- $Times(x) \iff x=28$
- $Plus(x) \iff x=29$
- $Succ(x) \iff x=30$
- $Logic(x) \iff x=31$
- $Div(x) \iff x=32$
- $Times(x) \iff x=33$
- $Plus(x) \iff x=34$
- $Succ(x) \iff x=35$
- $Logic(x) \iff x=36$
- $Div(x) \iff x=37$
- $Times(x) \iff x=38$
- $Plus(x) \iff x=39$
- $Succ(x) \iff x=40$
- $Logic(x) \iff x=41$
- $Div(x) \iff x=42$
- $Times(x) \iff x=43$
- $Plus(x) \iff x=44$
- $Succ(x) \iff x=45$
- $Logic(x) \iff x=46$
- $Div(x) \iff x=47$
- $Times(x) \iff x=48$
- $Plus(x) \iff x=49$

(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, $\overline{AxQuant}$ denotes the
constant $0$, $\text{LParen}$ denotes $0$, and $\text{LE}$ denotes $22$.

**Definition.** *Semiterm* and *Term* are unary predicates which are $\Delta_1^0$-defined in $S^1_2$ by the following inductive definition:

(a) $\neg\text{Semiterm}(0)$

(b) If $\text{Seq}(w)$ and $\text{Len}(w)=1$ and $\text{Var}(\beta(1,w))\text{Zero}(\beta(1,w))$ then $\text{Semiterm}(w)$.

(c) If $\text{Semiterm}(w)$ and $\text{Funct}(x)$ then $\text{Semiterm}(((\text{LParen}+x)\ast(w\text{RParen})))$.

(d) If $\text{Semiterm}(w)$, $\text{Semiterm}(v)$, and $\text{Funct}(x)$ then $\text{Semiterm}(((\text{LParen}+x)\ast(w\ast x)\ast(v\text{RParen})))$.

(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 $\Pi^1_1$-inductive definition.

A term is defined to be a semiterm without any bound variables:

$$\text{Term}(w) \iff \text{Semiterm}(w) \land (\forall z < \text{Len}(w) \neg \text{BVar}(\beta(Sz,w)))$$

We next define semiformalae and formalae. 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 formalae and atomic semiformalae by:

$$\text{ASemiFmla}(w) \iff \text{LParen}(\beta(1,w))\text{RParen}(\beta(\text{Len}(w),w)) \land (\exists x < \text{Len}(w)) (\text{Refl}(\beta(x,w)) \land$$

$$\land \text{Semiterm}(\text{Subseq}(w,x,2)) \land \text{Semiterm}(\text{Subseq}(w,x+1,\text{Len}(w))))$$

$$\text{AFmla}(w) \iff \text{ASemiFmla}(w) \land (\forall z < \text{Len}(w) \neg \text{BVar}(\beta(Sz,w)))$$

We also define what it means for a bound variable to appear bound in a semiformala:

$$\text{Free}(x,w) \iff \text{BVar}(x) \land \text{Seq}(w) \land (\forall i < \text{Len}(w) \land \forall i \neq i+1) (\text{Quant}(\beta(Si,w)) : x \neq \beta(i+1,w))$$

$$\text{Bound}(x,w) \iff \text{BVar}(x) \land \text{Seq}(w) \land \neg \text{Free}(x,w)$$

$$\text{Compat}(w,v) \iff \text{Seq}(v) \land \text{Seq}(w) \land (\forall z < \text{Len}(v)) \neg (\exists \beta(x,w) \land$$

$$\land (\forall z < \text{Len}(w)) \neg \text{Bound}(\beta(Sz,v),w))$$

Because of the way we have defined $\text{Bound}$ and $\text{Free}$ we will not allow semiformalae in which a
bound variable is both bound and free. For example, $(\forall x)(x \neq 0) \implies x \neq 0$ is not a valid semiformula.

We define $\text{SemiFmla}(w)$ by the following inductive definition:

(a) If $\text{ASemiFmla}(w)$ then $\text{SemiFmla}(w)$.

(b) If $\text{SemiFmla}(v)$ then $\text{SemiFmla}(\langle 0; \text{LParen}; \neg; \text{RParen} \rangle \ast (v \ast \text{RParen}))$.

(c) If $\text{SemiFmla}(v_1)$, $\text{SemiFmla}(v_2)$ and $\text{Conn}(2, z)$ and if $\text{Compat}(v_1, v_2)$ then $\text{SemiFmla}(\langle 0; \text{LParen}; v_1 \ast z; \ast; v_2 \ast \text{RParen} \rangle)$.

(d) If $\text{SemiFmla}(v_1)$, $\text{Quan}(x)$, $\text{BVar}(y)$, $\text{Semiterm}(v_2)$, $\text{Free}(y, v_1)$ and $\text{Compat}(v_1, v_2)$ and $(\forall u < \text{Len}(v_2))(\beta([u, v_2]) = y)$ then $\text{SemiFmla}(\langle 0; \text{LParen}; \text{LParen}; x; \ast; y; \ast; \text{LE}; \ast; v_2 \ast \text{RParen}; x; \ast; v_1 \ast \text{RParen} \rangle)$ and $\text{SemiFmla}(\langle 0; \text{LParen}; \text{LParen}; x; \ast; y; \ast; \text{RParen}; x; \ast; v_2 \ast \text{RParen}; x; \ast; v_1 \ast \text{RParen} \rangle)$.

(e) $\text{SemiFmla}(w)$ is true only as required by the above clauses.

We define $\text{Fmla}(w)$ to mean that $w$ codes a formula, that is to say, $w$ is a semiformula and no bound variable appears free in $w$:

$$\text{Fmla}(w) \iff \text{SemiFmla}(w) \land (\forall u < \text{Len}(w))(\beta([u, w]) = \text{Bound}(\beta([u, w])))$$

We next define how to count the alternation of bounded quantifiers in a formula. This allows $S^1_1$ to recognize $\Sigma^1_1$-formulas. We first must be able to distinguish sharply bounded from non-sharply bounded quantifiers. We define $\text{LTerm}(x)$ to be true iff $x$ codes a term of the form $[t]$

$$\text{LTerm}(x) \iff \text{Semiterm}(x) \land (\text{Len}(x) > 1 \ast \text{Log}([\beta(2, x)]))$$

$Q\text{Count}(w)$ is a function classifying the formula $w$ by its alternation of quantifiers.

$Q\text{Count}(w) < i, j >$ means $w \in \Sigma^1_{i+j}$, $Q\text{Count}(w) < 1, i >$ means $w \in \Pi^1_i$, and $Q\text{Count}(w) < 2, i >$ means $w \in \Pi^1_{i+1}$. $Q\text{Count}$ is defined by the following $\beta$-inductive definition:

(a) If $\neg \text{SemiFmla}(w)$ then $Q\text{Count}(w) = 0$.

(b) If $\text{ASemiFmla}(w)$ then $Q\text{Count}(w) = 2, 0 >$.

(c) If $w = \langle 0; \text{LParen}; \neg; \text{RParen} \rangle$ then...
\[ Q\text{Count}(w) = \begin{cases} <0, i> & \text{if } Q\text{Count}(v) = <1, i> \\ <1, i> & \text{if } Q\text{Count}(v) = <0, i> \\ <2, i> & \text{if } Q\text{Count}(v) = <2, i> \\ 0 & \text{otherwise} \end{cases} \]

(d) Suppose \( w = (\text{Var}(x) \cdot \text{Var}(y) \cdot \text{Var}(z)) \cdot \text{Conn}(2, x), \text{SemiFmla}(v_1), \text{SemiFmla}(v_2) \) and \( \text{Comp}(v_1, v_2) \). If \( Q\text{Count}(v_1) = 0 \) or \( Q\text{Count}(v_2) = 0 \) then \( Q\text{Count}(w) = 0 \). Otherwise, define

\[ Q\text{Imp}(v_1) = \begin{cases} Q\text{Count}(v_1) & \text{if } \neg \text{Imp}(x); \beta(1, Q\text{Count}(v_1)) = 2 \\ <1, \beta(1, Q\text{Count}(v_1)), \beta(2, Q\text{Count}(v_1))> & \text{otherwise} \end{cases} \]

and let \( i_1, j_1, j_2, j_4 \) be so that \( Q\text{Imp}(v_1) = <i_1, j_1> \) and \( Q\text{Count}(v_2) = <i_2, j_2> \). Then

\[ Q\text{Count}(w) = \begin{cases} <i_1, j_1> & \text{if } j_2 < j_1 \land (i_2 = 2i_1 - j_2) \\ <i_2, j_2> & \text{if } j_2 > j_1 \land (i_1 = 2i_1 - j_2) \\ <i_1, j_1> & \text{if } j_2 = j_1 \land i_1 = i_2 \\ <2, i_1 + 1> & \text{otherwise} \end{cases} \]

(e) Suppose \( \text{SemiFmla}(w), \text{SemiFmla}(v_1), \text{SemiFmla}(v_2), \text{Quant}(z) \) and \( \text{BVar}(y) \) where \( w = (\text{Var}(x) \cdot \text{Var}(y) \cdot \text{Var}(x) \cdot \text{Var}(z) \cdot \text{Var}(y) \cdot \text{Var}(z) \cdot \text{Var}(y)) \). If \( Q\text{Count}(v_1) = 0 \) then \( Q\text{Count}(w) = 0 \). Otherwise define

\[ Q\text{Count}(w) = \begin{cases} Q\text{Count}(v_1) & \text{if } L\text{Term}(v_2) \land \beta(1, Q\text{Count}(v_1)) = \text{QType}(x) \\ <\text{QType}(x), \beta(2, Q\text{Count}(v_1))> & \text{if } \neg L\text{Term}(v_2) \land \beta(1, Q\text{Count}(v_1)) = 2 \\ <\text{QType}(x), 1 + \beta(2, Q\text{Count}(v_1))> & \text{otherwise} \end{cases} \]

where \( \text{QType}(x) = \begin{cases} 0 & \text{if } z = \text{EQuant} \\ 1 & \text{otherwise} \end{cases} \)

(f) If \( w = (\text{Var}(x) \cdot \text{Var}(y) \cdot \text{Var}(z) \cdot \text{Var}(y) \cdot \text{Var}(z) \cdot \text{Var}(y)) \) where \( \text{Quant}(z) \), \( \text{BVar}(y) \), and \( \text{SemiFmla}(v_1) \), then \( Q\text{Count}(w) = 0 \).

That completes the definition of \( Q\text{Count} \).
Another important operation we need to $\Sigma_1^1$-define in $S_2^1$ is the substitution of a term into a formula or term. First define

$$
\text{SubOK}(w,x,v) \iff (\forall v.(\forall \bar{x} \in \text{Var}(v) \cdot (\forall \bar{y} \in \text{Var}(x) \cdot \text{Free}(\bar{x},w)) \land \\
\lambda(S\text{emTerm}(w)) \land \\
\lambda(\forall i < \text{Len}(v))(\text{Bound}(	heta(S_i,v),v)))
$$

We define $\text{Sub}(w,x,v)$ to be the function satisfying:

$$
z = \text{Sub}(w,x,v) \iff \exists v.\forall \bar{u} \cdot \text{SubOK}(w,x,v) \land \\
\lambda(\forall \bar{u} < \text{Len}(v))(\text{Seq}(\bar{u}) \cdot \text{Seq}(v) \land \\
\lambda(\forall i < \text{Len}(z))(\theta(S_i,w) \equiv \theta(S_i,v) \\
\land \lambda(\forall i < \text{Len}(w))(\theta(S_i,w) \equiv \theta(S_i,v)) \\
\land \lambda(\forall i < \text{Len}(v))(\theta(S_i,w) \equiv \theta(S_i,v)) \\
\land (\theta(S_j,v) \equiv \theta(S_j,w)) \\
\land (\theta(S_j,v) \equiv \theta(S_j,w)) \\
\land (\theta(S_j,v) \equiv \theta(S_j,w)))
$$

so $\text{Sub}(w,x,v)$ is the result of substituting $v$ for $x$ in $w$. We leave to the reader the proof that $\text{Sub}$ is a $\Sigma_1^1$-defined function of $S_2^1$ (the existence and uniqueness conditions of the above defining equation must be proved in $S_2^1$.) We also claim that

$$
S_2^1 \vdash \text{Fmla}(w) \land \text{Term}(v) \land \text{SubOK}(w,x,v) \land \\
\lambda(\forall v \in \text{Len}(v))(\text{Seq}(\bar{u}) = \text{Seq}(v))
$$

This is proved by a p-inductive proof.

In addition to the $\text{Sub}$ function, we need a function for performing the simultaneous substitution of a term for a vector of variables. We define

$$
\text{VSubOK}(w,x,v) \iff \text{Seq}(\bar{u}) \land \text{Seq}(v) \land \text{Len}(x) = \text{Len}(v) \land \\
\lambda(\forall i < \text{Len}(z))(\text{SubOK}(w,\theta(i,v),\theta(i,v)) \land \\
\lambda(\forall i < \text{Len}(x))(\theta(i,\bar{u}) \equiv \theta(i,v) \\
\land \lambda(\forall i < \text{Len}(v))(\theta(i,\bar{u}) \equiv \theta(i,v)) \\
\land (\theta(i,\bar{u}) \equiv \theta(i,v)) \\
\land (\theta(i,\bar{u}) \equiv \theta(i,v)))
$$

So $\text{VSubOK}(w,x,v)$ is true if $x$ is a vector of distinct variables, $v$ is a vector of semiters, no variable in $x$ appears in any of the semiters in $v$ and if there are no bound variable conflicts which arise when the semiters of $x$ are substituted for the variables of $v$ in $w$. We can now define $\text{VSub}$ by:
\[ z = \text{VSub}(w, x, v) \iff (\neg \text{VSubOK}(w, x, v) \land x = 0) \lor \]

\[ \lor (\text{VSubOK}(w, x, v) \land \text{Unique}(z)) \land \]

\[ (\exists y \leq \text{SeqId}(w \# x, v))(\exists y \leq \text{Len}(y) + 1 \land \]

\[ (\forall i < \text{Len}(x))(v(i + 1, y) = \text{Sub}(i + 1, y), v(i + 1, y), v(i + 1, y))) \lor \]

\[ (v(1, y) = \text{Sub}(\text{Len}(x), v(1), y))) \]

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) Cendent(0) (this is the empty cedent).

(b) If Fmla(w) then Cendent(w).

(c) If Fmla(v1), Cendent(v2) and v2 \neq 0 then Cendent((v1 \cdot \text{Separ}) \# v2).

(d) Cendent(w) holds only as required by clauses (a)–(c).

Next we define a couple of functions for manipulating cedents:

\[ \text{CendentLen}(w) = \begin{cases} 0 & \text{if } \text{Len}(w) = 0 \\ 1 + (\# i < \text{Len}(w)) \cdot \text{Separ}(\beta(Si, w)) & \text{otherwise} \end{cases} \]

\[ \text{Cendent}((a, w), v) = v \iff (v = \text{CendentLen}(w))(a = 0) \lor \neg \text{Cendent}(w) \land \]

\[ (\exists y \leq \text{Len}(w))(\exists y \leq \text{Len}(w))(x = (\# i < \text{Len}(w))(a = 1 + (\# j < \text{Len}(w)) \cdot \text{Separ}(\beta(Sj, w))) \land \]

\[ (a = (\# i < \text{Len}(w))) \land (\text{Subseq}(w, Sj, Sj)) \land \]

\[ (\text{Subseq}(w, Sj, Sj)) \]

So Cendent((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 formule in w. Sequents are defined by

\[ \text{Sequent}(w) = (\exists u < \text{Len}(w))(\text{Arrow}(\beta(Su, w)) \land \text{Cendent}(\text{Subseq}(w, 1, Su))) \land \]

\[ (\text{Arrow}(w) = 1 + (\# i < \text{Len}(w)) \cdot \text{Arrow}(\beta(i + 1, w))) \]

\[ \text{Antecedent}(w) = \begin{cases} 0 & \text{if } \neg \text{Sequent}(w) \\ \text{Subseq}(w, 1, \text{Arrow}(w)) & \text{otherwise} \end{cases} \]

\[ \text{Succedent}(w) = \begin{cases} 0 & \text{if } \neg \text{Sequent}(w) \\ \text{Subseq}(w, 1 + \text{Arrow}(w), 1 + \text{Len}(w)) & \text{otherwise} \end{cases} \]
Gödel Incompleteness Theorems

We define $Q\text{Class}$ and $Q\text{Bded}$ as a function and predicate which count the number of alternations of quantifiers in sequents (and later is proofs). They are defined $\Pi^0_2$-inductively by:

(a) If $Fmla(w)$ then

$$Q\text{Class}(w) = S(2, Q\text{Count}(w))$$
$$Q\text{Bded}(w) \iff Q\text{Count}(w) \neq 0$$

(b) If $C\text{edent}(u)$ and $w = (v, S\text{epr}_v) = v_0$, then

$$Q\text{Class}(w) = \max(Q\text{Class}(v_1), Q\text{Class}(v_2))$$
$$Q\text{Bded}(w) \iff Q\text{Bded}(v_1) \land Q\text{Bded}(v_2)$$

(c) If $S\text{equent}(u)$ then

$$Q\text{Class}(w) = \max(Q\text{Class}(\text{Antecedent}(u)), Q\text{Class}(\text{Succedent}(u)))$$
$$Q\text{Bded}(w) \iff Q\text{Bded}(\text{Antecedent}(u)) \land Q\text{Bded}(\text{Succedent}(u))$$

So $Q\text{Bded}(w)$ is true if $w$ includes no unbounded quantifiers. $Q\text{Class}(w)$ is equal to the least $i$ such that every formula in $w$ is either a $\Sigma^0_i$ or a $\Pi^0_i$-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 $L\text{Axiom}(<x, w>)$ be a predicate defined to be true if $w = <0, w>$ where $w$ is a logical axiom of one of the following forms:

(a) $\Delta \rightarrow A$ where $A$ is an atomic formula.

(b) $\neg \rightarrow t = t$ where $t$ is any term.

(c) $t = s \rightarrow f(t) = f(s)$ where $s$ and $t$ are terms and $f$ is one of the functions $S$, $\llbracket x \rrbracket$, or $|x|_t$.

(d) $t_1 = s_1, t_2 = s_2 \rightarrow f(t_1, t_2) = f(s_1, s_2)$ where each $s_i$ and $t_i$ is a term and $f$ is one of the functions $\dotplus, \dotminus$, or $\#$.

(e) $t_1 = s_1, t_2 = s_2 \rightarrow f(t_1, t_2) = f(s_1, s_2)$ where each $s_i$ and $t_i$ is a term and $f$ is one of the relations $\leq$ or $=$. Let $\alpha$ be any unary $\Sigma^k_1$-definable function of $S^1_k$. We use $\alpha$ to enumerate a list of non-logical
axioms and define $\text{NLAxiom}_s(v)$ to be true iff (1) $v=v_0,\ldots,v_n>\gamma_i$ or (2) either $v_3$ is the Gödel number of one of the finite number of $\text{BASIC}$ axioms or of $v_2=v_3$ and (3) the following four conditions hold: (a) $p_i=\langle x_1,\ldots,x_n\rangle$ and $E_{V1}(x_i)$ and $\text{Term}(y_i)$, (b) $v_4=\text{VSub}(v_0,v_0,v_1)$, and (c) $v_4=\text{Fmla}(v_4)$. Thus the non-logical axioms are instances of formuale from $\text{BASIC}$ axioms or formulae in the range of $\alpha$. 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 $\alpha$ 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 $\text{BASIC}$ 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 $\Gamma_n\rightarrow\Delta_n$ of $P$. The sons of a node $n$ of $w$ correspond to the upper sequents of the inference in $P$ which yielded $\Gamma_n\rightarrow\Delta_n$. Accordingly, the valence of each node of $w$ is not greater than two. The label on each node of $w$ is $\langle x_{v_0},v_0\rangle$ where $v_0$ is a Gödel number of the sequent $\Gamma_n\rightarrow\Delta_n$ and $x_{v_0}$ is a code detailing the inference used to derive that sequent. We already explained in detail what $x_{v_0}$ is for initial sequents. For non-initial sequents, it suffices to take $x_{v_0}\leq 23$ to be equal to the number of the inference as described in Chapter 4 or $x_{v_0}=24$ 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.

$\text{Proof}_s(w) \iff \text{"}w\text{ codes a proof with non-logical axioms specified by } \text{NLAxiom}_s\text{ and all inductions in } w\text{ are } \Delta^2_s\text{-PIND } s\text{.} \text{"}$

$\text{ProofFC}^s_s(w) \iff \text{"}\text{Proof}_s(w) \text{ and there are no free cuts in } w\text{.} \text{"}$

$\text{QBound}(w) \iff \text{"}\text{All quantifiers in } w\text{ are bounded.} \text{"}$

$\text{QClass}(w) \iff \text{"}i\text{ is the least number such that } w\text{ contains formulae in } w\text{ are in } \Sigma^b_i\text{ or } \Pi^b_i. \text{"}$

$\text{FreeForm}(w,0,i) \iff \text{"}i\text{-th formula of the antecedent of the endsequent of } w\text{ is } \text{ a free formula.}\text{"}$

$\text{FreeForm}(w,1,i) \iff \text{"}i\text{-th formula of the succedent of the endsequent of } w\text{ is } \text{ a free formula.}\text{"}$

$i=\text{INDType}(w) \iff \text{"}$\exists i\geq 6\text{ such that } \text{all induction inferences in } w\text{ are } \Sigma^b_{i+1}\text{-PIND inferences, or } i=0\text{ and there are no induction inferences in } w.\text{"}$

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:

\[
\begin{align*}
\text{ProofFQ}_i^j & \iff \text{Proof}_i^j(w) \land \text{QClass}(w) \leq i \land \text{INDType}(w) \leq i+1 \\
\text{ProofFF}_i^j & \iff \text{Proof}_i^j(w) \land \text{QClass}(w) \leq i \land \text{INDType}(w) \leq i+1 \\
\text{ProofBD}_i^j (w) & \iff \text{Proof}_i^j(w) \land \text{QClass}(w) \leq i \land \text{INDType}(w) \leq i+1 \\
\text{ProofFC}_i^j (w) & \iff \text{Proof}_i^j(w) \land \text{QClass}(w) \leq i \land \text{INDType}(w) \leq i+1 \\
\text{ProofBR}_i^j (w) & \iff \text{Proof}_i^j(w) \land \text{QClass}(w) \leq i \land \text{INDType}(w) \leq i+1
\end{align*}
\]

When we use \( \sigma \) as a subscript, it denotes any function with range contained in the set of Gödel numbers of \textit{BASIC} axioms. Thus \text{ProofBD}_i(w), \text{ProofBQ}_i(w) and \text{ProofFC}_i(w) each imply that \( w \) is a proof in the theory \( S_i^2 \). Also, \text{ProofBD}_i^j(w), \text{ProofBQ}_i^j(w) and \text{ProofFC}_i^j(w) mean that \( w \) has no induction inferences at all. The difference between \text{ProofBD}_i(w) and \text{ProofBQ}_i(w) is that \text{ProofBD}_i(w) means that \( w \) codes a bounded \( S_i^2 \)-proof whereas \text{ProofBQ}_i(w) means that \( w \) codes a bounded \( S_i^2 \)-proof and that all the formulae in \( w \) are \( \Sigma_i^1 \) - or \( \Pi_i^1 \)-formulae.

Define the function \text{EndSequent}(w) to be \( \beta(2, 	ext{Root}(w)) \). Also define \text{Prf} as

\[
\text{Prf}(w,a) \iff a = \text{EndSequent}(w).
\]

So \( \text{Prf}(w,a) \) is true iff \( a \) is the Gödel number of the sequent or formula proved by the proof \( w \). We further define:

\[
\begin{align*}
\text{Prf}_i^j (w,v) & \iff \text{Prf}_i^j(w) \land \text{Prf}(w,v) \\
\text{PrfFC}_i^j (w,v) & \iff \text{PrfFC}_i^j(w) \land \text{Prf}(w,v) \\
\text{PrfBF}_i^j (w,v) & \iff \text{PrfBF}_i^j(w) \land \text{Prf}(w,v) \\
\text{PrfBD}_i^j (w,v) & \iff \text{PrfBD}_i^j(w) \land \text{Prf}(w,v) \\
\end{align*}
\]

\[
\begin{align*}
\text{ThmFC}_i^j (v) & \iff \exists w \exists \text{Prf}_i^j (w,v) \\
\text{ThmBB}_i^j (v) & \iff \exists w \exists \text{PrfBD}_i^j (w,v) \\
\end{align*}
\]

The last nine predicates are definitely not \( \Delta_i^1 \) with respect to 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: \( \text{Num}(a) \) is a function \( \Sigma^1 \)-defined in \( S^1_2 \) so that \( \text{Num}(x) \) is the Gödel number of the term \( I_a \). We know that \( \text{Num}(a) \) can be \( \Sigma^1 \)-defined in \( S^1_2 \) since it is easy to give a \( p \)-inductive definition for \( \text{Num} \).

From now on we will use \( \bar{\gamma} \) to denote the Gödel number of a term, formula, sequent, or proof \( \bar{x} \). If \( n \in \mathbb{N} \) then \( \bar{\gamma}^n \) denotes \( \bar{I}_a \) or \( \text{Num}(a) \).

We will write \( FSub(\bar{A}, \bar{a}, t) \) to mean \( \text{Sub}(\bar{A}, \bar{a}, \text{Num}(t)) \); in other words, \( FSub(\bar{A}, \bar{a}, t) \) is the formula obtained by replacing all occurrences of the free variable \( a \) in the formula \( \bar{A} \) by the term \( I_a \). If \( \bar{a} \) is an \( n \)-tuple of free variables and \( \bar{x} \) is an \( n \)-tuple of terms then we write

\[ FSub(\bar{A}, \bar{a}, \bar{x}) \]

as an abbreviation for

\[ FSub(\ldots (FSub(\bar{A}, \bar{a}_1, \bar{x}_1) \ldots , \bar{a}_n, \bar{x}_n). \]

To improve readability, we shall frequently use \( FSub \) implicitly in the following way. Let \( \bar{A}(a_1, \ldots, a_k) \) be a formula. Then \( \bar{A}(I_{a_1}, \ldots, I_{a_k}) \) is an abbreviation for \( FSub(\bar{A}(\bar{a}), \bar{a}, \bar{x}) \). For example, we shall write

\[ S^1_2 \vdash \text{ThmBD}(\bar{y}) \bar{A}(I_{a_1}, \ldots, I_{a_k}) \]

as an abbreviation for

\[ S^1_2 \vdash \text{ThmBD}(\bar{y}) FSub(\bar{A}(a_1, \ldots, a_k), \bar{a}, \bar{x}). \]

The next theorem is very important for establishing the Gödel incompleteness theorems.

Theorem 4:
(a) Let \( \bar{A} \) be any \( \Sigma^1 \)-formula in the language of Bounded Arithmetic. Let \( a_1, \ldots, a_k \) be all the free variables of \( \bar{A} \). Then there is a term \( t_{\bar{a}}(\bar{a}) \) such that

\[ S^1_2 \vdash \bar{A}(\bar{a}) \land \exists \bar{a}. \text{FPC}(\bar{a}, \bar{a}, \bar{x}). \]

\[ S^1_2 \vdash \bar{A}(\bar{a}) \land \exists \bar{a}. \text{FPC}(\bar{a}, \bar{a}, \bar{x}) \].
(b) Let \( A \) be of the form \((\exists x)B(x,x)\) where \( B \) is a \( \Sigma^1_1 \)-formula in the language of Bounded Arithmetic. Let \( a_0, \ldots, a_p \) be all the free variables of \( A \). Then

\[
S^1_2 \vdash A(\bar{x}) \supset \text{ThmFCF}^{(1)}(\forall \bar{y}, FS_{\text{Sub}}(\bar{A}(\bar{x}), \bar{y})).
\]

So Theorem 4 asserts that for any \( \Sigma^1_1 \)-formula \( A(\bar{x}) \), \( S^1_2 \) proves that for all values \( \bar{y} \) such that \( A(\bar{y}) \) is true there is an induction free, free cut free proof of \( A(a_0, \ldots, a_p) \).

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 \( a_1, \ldots, a_n \). Then

\[
S^1_2 \vdash \text{ThmFCF}^{(1)}(\forall \bar{y}(I_{a_1}, \ldots, I_{a_n}) = I_{I(\bar{a}_1), \ldots, \bar{a}_n}).
\]

**Proof:** by induction on the complexity of \( t \).

(a) Suppose \( t \) is the constant term 0. Then \( S^1_2 \vdash \text{ThmFCF}^{(1)}(\forall \bar{y}(0 = 0)) \) is immediate from the equality axioms.

(b) Suppose \( t \) is a variable symbol \( a \). Then \( S^1_2 \vdash \text{ThmFCF}^{(1)}(\forall \bar{y}(I_{a} = I_{a})) \) is immediate from the equality axioms.

(c) Suppose \( t \) is \( S(r) \). By the induction hypothesis, \( S^1_2 \) proves that for all \( \bar{y} \) there exists a proof that \( r(I_{a_1}, \ldots, I_{a_n}) = I_{I(\bar{a}_1), \ldots, \bar{a}_n}) \). So it suffices to show that

\[
S^1_2 \vdash \text{ThmFCF}^{(1)}(\forall \bar{y}(S(I_{a_1}) = I_{I(\bar{a}_1)})).
\]

This is proved by \( \Sigma^1_1 \)-PIND with respect to \( b \). Since there is a proof of \( S(I_b) = I_b \) it is clearly true for \( b = 0 \). To deal with the induction step, we argue informally inside \( S^1_2 \). The induction hypothesis is that there is a free cut free \( S^1_2 \)-proof of \( S(I_{a_1}) = I_{S(I_{a_1})} \). We divide the argument into two cases. First, suppose \( b \) is even. Then \( S^1_2 \) proves immediately that \( S(I_b) = I_b + 50 \) and since \( I_b + 50 \) is identical to \( I_b \) this case is done. Second, suppose \( b \) is odd. Then \( S^1_2 \) proves immediately that \( S(I_{a_1}) = 2(I_{a_1} + 2) = 2(S(I_{a_1})) \) and by combining that proof with the proof of \( S(I_{a_1}) = I_{S(I_{a_1})} \) we obtain, by an essential cut, a proof of \( S(I_b) = I_b \).

To apply \( \Sigma^1_1 \)-PIND we must find a uniform bound \( t_\delta \) so that the proof of \( S(I_b) = I_{\delta b} \) is coded by a Gödel number \( t_{\delta b}(b) \). This is readily done, since in either case of the argument for the induction step, the difference in size of the proof of \( S(I_b) = I_{\delta b} \) over the size of the proof of \( S(I_{a_1}) = I_{S(I_{a_1})} \) is bounded by an amount proportional to the size \( |b| \) of \( b \). Thus the size of the free cut free \( S^1_2 \)-proof of \( S(I_b) = I_{\delta b} \) is quadratic in the size of \( b \).
(d) Suppose \( t \) is \( r + s \). As in (c), it will suffice to show that
\[
S^2_1 \vdash \text{ThmFCF}^{(\mathcal{C})}(\mathcal{I}_t + \mathcal{I}_c = I_{t + c}).
\]
Let \( \lambda \) and \( \kappa \) abbreviate \( \text{MSP}(b,u) \) and \( \text{MSP}(c,u) \) respectively. Let \( D(u) \) be the Gödel number of the formula
\[
I_\lambda + I_\kappa = I_{\lambda + \kappa}.
\]
We will show that
\[
S^2_1 \vdash \text{ThmFCF}^{(\mathcal{C})}(\mathcal{D}(\min(|b|,|c|)))
\]
and
\[
S^2_1 \vdash \text{ThmFCF}^{(\mathcal{C})}(\mathcal{D}(u - 1)).
\]
Then \( S^2_1 \) can use \( \Sigma^0_1 \text{-LIND} \) to conclude \( \text{ThmFCF}^{(\mathcal{C})}(\mathcal{D}(0)) \), which is what we need to show, as \( b = b' \) and \( c = c' \).

We argue informally inside \( S^2_1 \). Let \( v = \min(|b|,|c|) \); we want to show that \( S^2_1 \) proves \( D(u) \). Suppose without loss of generality that \( v = |c| \). Then \( c_v = 0 \), so \( D(0) \) is \( \mathcal{I}_{t + 0} = \mathcal{I}_{t + 0} \), and this is easily proved in \( S^2_1 \) by an equality axiom as \( I_{t + 0} \) and \( I_{t + 0} \) are the same term. We next argue the induction step. The induction hypothesis is that there is a free cut free \( S^2_1 \text{-proof} \) of \( I_{t + \lambda} + I_{t + \kappa} = I_{t + \lambda + \kappa} \), and that \( u > 0 \). We want to show that there is an \( S^2_1 \text{-proof} \) of \( I_{t + \lambda + 1} + I_{t + \kappa + 1} = I_{t + \lambda + \kappa + 1} \). Note that \( b' = \lfloor b + 1 \rfloor \) and \( c_v = \lfloor c_v + 1 \rfloor \). There are two cases to consider. First suppose that one of \( b_{\lambda + 1} \) and \( c_{\lambda + 1} \) 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 \( I_{t + \lambda} + I_{t + \kappa} = I_{t + \lambda + \kappa} \) to get a proof of \( I_{t + \lambda + 1} + I_{t + \kappa + 1} = I_{t + \lambda + \kappa + 1} \). Second, suppose that both \( b_{\lambda + 1} \) and \( c_{\lambda + 1} \) are odd. Then \( S^2_1 \) can prove immediately from the BASIC axioms that \( I_{t + \lambda + 1} + I_{t + \kappa + 1} = 2(I_{t + \lambda + \kappa + 1}) \). We combine that with the \( S^2_1 \text{-proof} \) of \( I_{t + \lambda} + I_{t + \kappa} = I_{t + \lambda + \kappa} \) using an inessential cut to get an \( S^2_1 \text{-proof} \) of \( I_{t + \lambda + 1} + I_{t + \kappa + 1} = 2(I_{t + \lambda + \kappa + 1}) \). By (c), there is an \( S^2_1 \text{-proof} \) of \( I_{t + \lambda + 1} + 1 = I_{t + \lambda + \kappa + 1} + 1 \). From this we can use another inessential cut to obtain an \( S^2_1 \text{-proof} \) of \( I_{t + \lambda + 1} + I_{t + \kappa + 1} = 2(I_{t + \lambda + \kappa + 1}) \). Now we are done, since \( 2(I_{t + \lambda + \kappa + 1}) \) and \( I_{t + \lambda + \kappa} = I_{t + \lambda + \kappa} \) are the same term.

To apply \( \Sigma^0_1 \text{-LIND} \) to conclude that \( \text{ThmFCF}^{(\mathcal{C})}(\mathcal{D}(0)) \) we must find a term \( u \), which bounds the size of the \( S^2_1 \text{-proofs} \) constructed above. Because of the size bound established in (c), we know that the increase in size of the proof of \( D(u - 1) \) over the size of the proof of \( D(u) \) is bounded by an amount quadratic in the size of \( b + c \). Hence the size of the free cut free \( S^2_1 \text{-proof} \) of \( I_{t + \lambda + c} = I_{t + \kappa + c} \) is bounded by a cubic polynomial of the sizes \( |b| \) and \( |c| \) of \( b \) and \( c \).
(e) Suppose \( t \) is r-\( \#t \). As before, it suffices to show that \( S_2^2 \) proves

\[
\text{ThmFCF}^{<\text{I}}[1, \leftarrow 1, v, \leftarrow] \text{.}
\]

We shall prove this by using \( \Sigma_1^b \text{-PIND} \) with respect to the variable \( b \).

We argue informally inside \( S_2^2 \). First we consider the case \( b = 0 \); we want to show that \( S_2^2 \) proves \( I_0 \equiv I_0 \). This is easily proved in \( S_2^2 \) from the \( \text{BASIC} \) 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 \( S_2^2 \)-proof of \( I_{[1]} \equiv I_{[1]} \) and we want to show that there is an \( S_2^2 \)-proof of \( I_{[1]} \equiv I_{[1]} \) for all \( \alpha \). There are two cases. First, if \( b \) is even then \( I_{[1]} \) in \( I_{[1]} \equiv I_{[1]} \) and \( I_{[1]} \) is \( I_{[1]} \equiv I_{[1]} \). Hence the proof of \( I_{[1]} \equiv I_{[1]} \) is easily extended to a proof of \( I_{[1]} \equiv I_{[1]} \). Second, if \( b \) is odd then \( 2 \cdot [4] + 1 = b \) and \( S_2^2 \) can prove from the \( \text{BASIC} \) axioms that \( I_{[1]} \equiv I_{[1]} \equiv I_{[1]} \). We combine this with the proof of \( I_{[1]} \equiv I_{[1]} \equiv I_{[1]} \) using an inessential cut to get an \( S_2^2 \)-proposition of \( I_{[1]} \equiv I_{[1]} \). By (d), there is a free cut free \( S_2^2 \)-proof of \( I_{[1]} \equiv I_{[1]} \equiv I_{[1]} \), and we can use this and an inessential cut to get the desired \( S_2^2 \)-proposition of \( I_{[1]} \equiv I_{[1]} \), which completes the induction step.

Since we used (d) in the induction step argument, the size of the free cut free \( S_2^2 \)-proof of \( I_{[1]} \equiv I_{[1]} \equiv I_{[1]} \) constructed above is bounded by a quartic polynomial of the sizes \( |b| \) and \( |c| \) of \( b \) and \( c \).

(f) Suppose \( t \) is r-\( \#t \). It suffices to show that \( S_2^2 \) proves

\[
\text{ThmFCF}^{<\text{I}}[1, \leftarrow 1, v, \leftarrow] \text{.}
\]

First, it is clear that if \( b = 0 \) there is an \( S_2^2 \)-proposition of this using the \( \text{BASIC} \) axioms. We shall prove the case \( b > 0 \) in two parts. First, we show by \( \Sigma_1^b \text{-PIND} \) with respect to \( c \) that there is a free cut free \( S_2^2 \)-proposition of \( I_{[1]} \equiv I_{\leftarrow 1} \) for all \( c \); second, we use \( \Sigma_1^b \text{-PIND} \) with respect to \( b \) to prove that there is a free cut free \( S_2^2 \)-proposition of \( I_{[1]} \equiv I_{\leftarrow 1} \) for all \( b \) and \( c \). We shall argue informally inside \( S_2^2 \).

First, it is clear that there is an \( S_2^2 \)-proposition of \( I_{[1]} \equiv I_{\leftarrow 1} \) (since \( 1 \equiv 2 \)). Suppose there is a free cut free \( S_2^2 \)-proposition of \( I_{[1]} \equiv I_{\leftarrow 1} \) where \( \alpha > 1 \). From the \( \text{BASIC} \) axioms, \( S_2^2 \)-proposition of \( I_{[1]} \equiv I_{\leftarrow 1} \) where \( \alpha > 1 \), then there is a free cut free \( S_2^2 \)-proposition of \( I_{[1]} \equiv I_{\leftarrow 1} \) where \( \alpha > 1 \). But \( 2 \cdot [4] \equiv [4] \equiv 1 \) in the same way as \( I_{[1]} \equiv I_{[1]} \), and we are done.

Second, suppose there is a free cut free \( S_2^2 \)-proposition of \( I_{[1]} \equiv I_{\leftarrow 1} \) where \( \alpha > 2 \). From the \( \text{BASIC} \) axioms, \( S_2^2 \)-proposition of \( I_{[1]} \equiv I_{\leftarrow 1} \) where \( \alpha > 2 \). Hence, by (e), there is a free cut free \( S_2^2 \)-proposition of \( I_{[1]} \equiv I_{[1]} \).

The size of the free cut free \( S_2^2 \)-proposition constructed above is bounded by a fifth-order polynomial of the lengths \( |b| \) and \( |c| \) of \( b \) and \( c \).
(g) Suppose $t$ is $[a]_t$ or $t$ is $[a]$. It suffices to show that

\[ \text{Thm} {\mathcal{F}} \neg \neg \neg [I_{a_t}, I_{a}] \]

and

\[ \text{Thm} {\mathcal{F}} \neg \neg \neg [I_{a}, I_{a}] \].

These are easily proved by using $\Sigma_1^1$-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 $t = a$, $\neg t = a$, $t \leq a$, or $\neg t \leq a$. By Lemma 5, $S_2^2$ proves that $(t(I_a, \ldots, I_a)) = t(I_a, \ldots, I_a)$ and $s(I_a, \ldots, I_a) = s(I_a, \ldots, I_a)$. So it will suffice to show that $S_2^2$ proves the following four formulae:

\[ b = c \Rightarrow \text{Thm} {\mathcal{F}} \neg \neg \neg [I_b = I_c] \]
\[ \neg b = c \Rightarrow \text{Thm} {\mathcal{F}} \neg \neg \neg [\neg I_b = I_c] \]
\[ b \leq c \Rightarrow \text{Thm} {\mathcal{F}} \neg \neg \neg [I_b \leq I_c] \]
\[ \neg b \leq c \Rightarrow \text{Thm} {\mathcal{F}} \neg \neg \neg [\neg I_b \leq I_c] \]

These are readily proved by induction on the lengths of $b$ and $c$. The sizes of the free cut free $S_2^2$-proofs are bounded by a quadratic polynomial of the lengths $|b|$ and $|c|$ of $b$ and $c$.

(b) Suppose $A$ is $B(\exists x) \psi(\exists x)$ and that Theorem 4 has already been established for $B$ and $\psi$. Thus,

\[ S_2^2 \vdash B(a) \Rightarrow \exists x \leq t_B \Prf {\mathcal{F}} \neg \neg \neg [\exists x \leq t_B] \]

and

\[ S_2^2 \vdash \psi(\exists x) \Rightarrow \exists x \leq t_{\psi} \Prf {\mathcal{F}} \neg \neg \neg [\exists x \leq t_{\psi}] \]

But it is easy for $S_2^2$ to prove that, given such a proof $t_B$ or $t_{\psi}$, adding an (v-rewrite) inference gives a $S_2^2$-proof of $A(I_{a'}, \ldots, I_{a'})$. The bounding term $t_{\psi}$ is easily obtained from $t_B$ and $t_{\psi}$.
(c) Suppose $A$ is $B(x,C)$. The argument for this case is similar to the argument for (b).

(d) Suppose $A$ is $(\forall z \leq x)B(x,z)$. By the induction hypothesis, $S_1^t$ proves

$$B(x,y) \supset (\exists w \leq z)(z \in t(x, y) \land \forall z \leq x \exists w \leq \varepsilon(x, w)) \land F_{\text{Scr}}(\varepsilon(x, w), \varepsilon(x, x), x, y).$$

We let $t(x, y) = \varepsilon(x, w)$. Then by use of $\Sigma_1^t$-LIND with respect to $\varepsilon$, $S_1^t$ proves

$$A \land \varepsilon(x, w) \leq \varepsilon(x, w) \supset (\exists w \leq \varepsilon(x, w))F_{\text{Scr}}(\varepsilon(x, w), \varepsilon(x, x), x, y, \varepsilon(x, w)), \quad (\exists w \leq \varepsilon(x, w))F_{\text{Scr}}(\varepsilon(x, w), \varepsilon(x, x), x, y, \varepsilon(x, w)).$$

where $\varepsilon(x, w) = 2^\varepsilon(\gamma, t)$, where $\gamma$ is a suitable constant. This is because the proofs of $B(I_{a_0}, \ldots, I_{a_n}, I_k)$ for $b \leq a$ can be put together via inessential cut inferences to obtain a free cut free $S_1^t$-proof of $b \leq a$.

By Lemma 5, $S_2$ proves

$$A \land \exists w \leq \varepsilon(x, w) \supset (\exists w \leq \varepsilon(x, w))F_{\text{Scr}}(\varepsilon(x, w), \varepsilon(x, x), x, y, \varepsilon(x, w)).$$

(c) Suppose $A$ is $(\exists x \leq x)B(x,x)$. By the induction hypothesis, $S_1^t$ proves

$$B(x,y) \supset (\exists w \leq z)(z \in t(x, y) \land \forall z \leq x \exists w \leq \varepsilon(x, w)) \land F_{\text{Scr}}(\varepsilon(x, w), \varepsilon(x, x), x, y).$$

Then $S_1^t$ proves

$$A(x) \supset (\exists x \leq x)(\exists w \leq x)F_{\text{Scr}}(\varepsilon(x, w), \varepsilon(x, x), x, x).$$

We argue informally in $S_1^t$. Suppose $A(x)$. Then we have just shown that there are an $x \leq x$ and a $w$ so that $w$ codes a free cut free $S_1^t$-proof of $B(I_{a_0}, \ldots, I_{a_n}, I_k)$. By Case (a), there is a free cut free $S_1^t$-proof of $I_{a_0} \leq x(I_{a_0}, \ldots, I_{a_n})$. We can combine these two proofs using an inessential cut and a $(\exists x \leq x)$-inference to get a free cut free $S_1^t$-proof of

$$(\exists x \leq x)(I_{a_0}, \ldots, I_{a_n})B(I_{a_0}, \ldots, I_{a_n}, x)$$

which is what we needed to show.

(f) Suppose $A$ is $(\exists x)B(x,x)$. 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 \( \Sigma_1^1 \)-defined functions or \( \Delta_1^1 \)-defined predicates. But of course any \( \Delta_1^1 \)-formula \( A \) which may include \( \Sigma_1^1 \)-defined function symbols and \( \Delta_1^1 \)-defined predicate symbols is equivalent, provably in \( S_3^1 \), to two formulae \( A_E \) and \( A_H \) where \( A_E \) and \( A_H \) are in \( \Sigma_1^1 \) and \( \Pi_1^1 \) 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 \( \Delta_1^1 \)-formula of \( S_3^1 \). That is, there are \( \Sigma_1^1 \)-formulae \( A_1 \) and \( A_2 \) such that

\[
S_3^1 \vdash A \iff A_1 \iff A_2.
\]

Then,

\[
S_3^1 \vdash A(\overline{z}) \supset \text{ThmFCF}^{\xi}(\text{FSub}(\overline{[A_1],[\overline{z}],\overline{z}]}))
\]

\[
S_3^1 \vdash \neg A(\overline{z}) \supset \text{ThmFCF}^{\xi}(\text{FSub}(\overline{[\neg A_2],[\overline{z}],\overline{z}]})).
\]

(b) Let \( A \) be any \( \Delta_1^1 \)-formula of \( S_3^1 \). Then

\[
S_3^1 \vdash A(\overline{z}) \supset \text{ThmBD}^{\xi}(\text{FSub}(\overline{[A],[\overline{z}],\overline{z}}]))
\]

\[
S_3^1 \vdash \neg A(\overline{z}) \supset \text{ThmBD}^{\xi}(\text{FSub}(\overline{[\neg A],[\overline{z}],\overline{z}}))).
\]

Note that in (b), \( \text{ThmBD}^{\xi} \) is used instead of \( \text{ThmFCF}^{\xi} \). Unlike Theorem 4 and Lemma 5, Corollary 6(b) would still hold if we enlarged the syntax of our metamathematics to include symbols for \( \Sigma_1^1 \)-defined functions and \( \Delta_1^1 \)-defined predicates.

**Corollary 7:** Let \( A(k) \) be one of the formulae \( \text{Thm}(\xi), \text{ThmBQ}^{\xi}(\xi), \text{ThmBD}^{\xi}(\xi), \text{ThmBD}^{\xi}(\xi), \) etc. Then

\[
S_3^1 \vdash (\exists z)[A(x) \supset \text{ThmBD}^{\xi}(\text{FSub}(\overline{[A],[\overline{z}],[\overline{x}]}))].
\]

7.5. G"odel 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 \( \text{ThmFCF} \) instead of \( \text{Thm} \); 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 $e$. Then there is a sentence $\phi$ such that

$$S^2_2 \vdash \phi \leftrightarrow \psi(\overline{\phi})$$

Furthermore, if $\phi$ is a $\Pi^0_2$-formula, then so is $\phi$. If $\psi$ is provably equivalent to a $\Sigma^1_2$-formula (resp. $\Pi^1_2$-formula) then so is $\phi$.

Proof: Since $\text{Sub}$ and $\text{Num}$ are $\Sigma^1_2$-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 $\psi(\text{Sub}(a, \overline{a}, \text{Num}(a)))$ such that if $\psi$ is a $\Sigma^1_2$- (respectively, $\Pi^1_2$-) formula then so is $\chi$. Define $\phi$ to be the sentence $\chi(\overline{\chi})$. So

$$S^2_1 \vdash \phi \leftrightarrow \psi(\text{Sub}(\overline{\chi}, \overline{a}, \text{Num}(\overline{\chi})))$$

By the definition of $\phi$ and the results of §7.4, we certainly have

$$S^2_1 \vdash \phi \leftrightarrow \psi(\overline{\phi})$$

which shows that $S^2_1 \vdash \phi \leftrightarrow \psi(\overline{\phi})$.

The fact that the quantifier structure of $\phi$ is the same as that of $\psi$ is immediate from the fact that $\phi$ is a substitution instance of $\psi$ and from Theorem 2.2. □

For added generality, we will work in theories stronger than $S^2_1$.

Definition: Let $\alpha$ be a unary $\Sigma^1_2$-defined function of $S^2_1$. We define $S^2_{1,\alpha}$ to be the theory such that

(a) The language of $S^2_{1,\alpha}$ is the language of Bounded Arithmetic.
(b) The axioms of $S^2_{1,\alpha}$ are the BASIC axioms plus all formulae with Gödel number in the range of $\alpha$.
(c) $S^2_{1,\alpha}$ has all the $\Sigma^1_2-PIND$ inference rules.

Example: Let $PA$ be Peano Arithmetic. Define $\beta$ by

$$\beta(n) = \begin{cases} \overline{a-a} & \text{if } n \text{ is not a Gödel number of a } PA \text{-proof} \\ m & \text{if } n \text{ codes a } PA \text{-proof of the sequent with Gödel number } m \end{cases}$$

Then $S_{2,\beta}$ is equivalent to $PA$. 
Definition: Let $\alpha$ be as above and fix $i \geq 1$. Since $Prf_{FCF}^i$ is $\Delta^i_1$ with respect to $S^i_1$, we can choose some formula $A(\forall \Pi^1_0$ such that $S^i_2 \vdash A(w,a) \rightarrow \neg Prf_{FCF}^i(w,a)$. Now let $\psi$ be the formula $(\forall w)A(w,a)$. Define $\phi^i_\psi$ to be the formula whose existence is guaranteed by Lemma 8 such that

$$S^i_2 \vdash \phi^i_\psi \rightarrow \psi(\phi^i_\psi).$$

Note that $\phi^i_\psi$ is a $\Pi^i_1$-formula of the form $(\forall w)B$ where $B$ is a $\Pi^i_1$-formula which is $\Delta^i_1$ with respect to $S^i_1$. Also,

$$S^i_2 \vdash \phi^i_\psi \rightarrow \neg Thm_{FCP}^i[\phi^i_\psi].$$

Theorem 9: (Gödel's First Incompleteness Theorem). Let $\omega$, $S^i_1$, and $\phi^i_\psi$ be as above, with $i \geq 1$. Suppose $S^i_1$ is consistent. Then,

$$S^i_1 \vdash \psi(\phi^i_\psi).$$

Proof: (by contradiction). Suppose $S^i_1 \vdash \neg \phi^i_\psi$. Then by the cut elimination theorem (Theorem 4.3), there is a free cut free $S^i_1$-proof of $\phi^i_\psi$. Hence, by Corollary 7,

$$S^i_2 \vdash Thm_{FCP}^i[\phi^i_\psi].$$

From the assumption that $S^i_1 \vdash \phi^i_\psi$ and the definition of $\phi^i_\psi$,

$$S^i_2 \vdash \neg Thm_{FCP}^i[\phi^i_\psi],$$

and since $S^i_1 \supseteq S^i_2$, this contradicts the consistency of $S^i_1$.

Q.E.D. □

Definition: The following predicates assert the consistency of various natural deduction proof systems:

- $Con^i_\omega$ \iff $\neg Thm_{\omega}[\rightarrow \rightarrow]$.
- $Con_{BQ}^i$ \iff $\neg Thm_{BQ}^i[\rightarrow \rightarrow]$.
- $Con_{BQ'}^i$ \iff $\neg Thm_{BQ'}^i[\rightarrow \rightarrow]$.
- $Con_{BD}^i$ \iff $\neg Thm_{BD}^i[\rightarrow \rightarrow]$.
- $Con_{BD'}^i$ \iff $\neg Thm_{BD'}^i[\rightarrow \rightarrow]$.
Gödel Incompleteness Theorems

\[ \text{Con}BD \iff \neg \text{Thm}BD^! \]
\[ \text{Con}FCF \iff (\exists x \text{Thm}FCF^! x \land \text{Thm}FCF^! (0+\text{LParen}+\text{Nat}\leftrightarrow (x+\text{RParen}))) \]
\[ \text{Con}FCF^! \iff (\exists x \text{Thm}FCF^! x \land \text{Thm}FCF^! (0+\text{LParen}+\text{Nat}\leftrightarrow (x+\text{RParen}))) \]

For example, \( \text{Con}FCF^! \) asserts that there is no formula \( A \) such that both \( A \) and \( \neg A \) have free cut free \( S^2_1 \)-proofs. It is necessary for our purposes that we define \( \text{Con}FCF \) 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 \( \text{Con}(R), \text{BDCon}(R), \text{BQCon}(R) \) and \( \text{FCFCCon}(R) \) to denote formulae expressing various consistency properties of \( R \). Thus, for example, we have:

\[
\begin{align*}
\text{Con}(S^2_1) & \iff \text{Con}^! \\
\text{Con}(S^2_{1\text{a}}) & \iff \text{Con}^! \\
\text{BQCon}(S^2_1) & \iff \text{BQCon}^! \\
\text{BQCon}(S^2_{1\text{a}}) & \iff \text{BQCon}^! \\
\text{BDCon}(S^2_1) & \iff \text{BDCon}^! \\
\text{BDCon}(S^2_{1\text{a}}) & \iff \text{BDCon}^! \\
\text{FCFCCon}(S^2_1) & \iff \text{FCFCCon}^! \\
\text{FCFCCon}(S^2_{1\text{a}}) & \iff \text{FCFCCon}^! \\
\end{align*}
\]

More generally, when \( R \) is any axiomatizable theory such that \( R \supseteq S^2_1 \), let \( \alpha \) be a \( \Sigma^1_1 \)-defined function of \( S^2_1 \) such that the range of \( \alpha \) is equal to the set of Gödel numbers of theorems of \( R \). Then \( \text{Con}(R), \text{BDCon}(R) \) and \( \text{FCFCCon}(R) \) are defined to be \( \text{Con}(S^2_{1\alpha}), \text{BDCon}(S^2_{1\alpha}) \) and \( \text{FCFCCon}(S^2_{1\alpha}) \), respectively.

In addition, the formulae \( \text{Prf}_R, \text{Thm}_R, \text{Prf}BD_R \) and \( \text{Prf}FCF_R \) will be used as alternative names for the formulae \( \text{Prf}^!, \text{Thm}^!, \text{Prf}BD^!, \) and \( \text{Prf}FCF^! \), respectively.

**Theorem 10:** (Gödel’s Second Incompleteness Theorem). Let \( \alpha, \phi^*_1 \) and \( S^2_{1\alpha} \) be as above, with \( \alpha \supseteq 1 \). Then,

\[ S^2_1 \vdash \neg \phi^*_1 \neg \neg \text{FCFCCon}(S^2_{1\alpha}) \]

and hence, if \( S^2_{1\alpha} \) is consistent,

\[ S^2_{1\alpha} \vdash \text{FCFCCon}(S^2_{1\alpha}) \].
Proof: Because \( \phi_4^i \) is a \( \Pi^0_2 \)-formula of the form \((\forall w)A\) where \( A \) is a \( \Pi^0_1 \)-formula which is \( \Delta^0_1 \) with respect to \( S_1 \), we have by Theorem 4 that

\[ S_2^1 \vdash \neg \phi_4^i \supset \text{Thm}_{\text{FCF}}^i(\neg \phi_4^i). \]

Also, by the definition of \( \phi_4^i \),

\[ S_2^1 \vdash \neg \phi_4^i \supset \text{Thm}_{\text{FCF}}^i(\overline{\neg \phi_4^i}). \]

It is also immediate from the definitions that

\[ S_2^1 \vdash \text{Thm}_{\text{FCF}}^i(\overline{\neg \phi_4^i}) \supset \text{Thm}_{\text{FCF}}^i(\overline{\neg \phi_4^i}). \]

Putting these three formulae together, we get, from the definition of \( \text{FCF} \text{Con}(S_2^1) \), that

\[ S_2^1 \vdash \neg \phi_4^i \supset \neg \text{FCF} \text{Con}(S_2^1). \]

Thus,

\[ S_2^1 \vdash \text{FCF} \text{Con}(S_2^1) \supset \phi_4^i. \]

By the First Incompleteness Theorem, \( S_2^1 \not\vdash \phi_4^i \), and hence

\[ S_2^1 \not\vdash \text{FCF} \text{Con}(S_2^1). \]

Q.E.D.

In Theorem 10 we only proved that \( S_2^1 \vdash \text{Con}_{\text{FCF}}^i \supset \phi_4^i \); we did not prove that \( S_2^1 \vdash \phi_4^i \supset \text{Con}_{\text{FCF}}^i \). In the standard treatments of Gödel's incompleteness theorem, \( \text{Con} \) is used instead of \( \text{Con}_{\text{FCF}} \). Then if \( \phi_4^i \) is defined using \( \text{Thm}^i \) in the same way that \( \phi^i \) was defined from \( \text{Thm}_{\text{FCF}}^i \), we have

\[ S_2^1 \vdash \neg \phi_4^i \iff \text{Con} \]

(see Theorem 5.6 of Feferman [9]). However, the author doubts that it is true that

\[ S_2^1 \vdash \phi_4^i \supset \text{Con}_{\text{FCF}}^i \]

since we are only considering free cut free proofs.
Since a free cut-free proof with bounded initial sequents and bounded consequent is a bounded proof, we have the following immediate corollary to the Second Incompleteness theorem.

Corollary 11: (i$\geq$1)
(a) $S_2^1 \vdash \text{FCFCon}(S_2^1)$
(b) $S_2^i \vdash \text{BDCon}(S_2^i)$
(c) $S_2^{i+1} \vdash \text{Cons}(S_2^i)$
(d) If all axioms of $S_2^{i+1}$ are bounded, then $S_2^{i+1} \vdash \text{BQCon}(S_2^{i+1})$
(e) If all axioms of $S_2^{i+1}$ are bounded, then $S_2^{i+1} \vdash \text{BDCon}(S_2^{i+1})$

Corollary 12: (Gödel)
(a) Let $PA$ be Peano arithmetic. Then $PA \nvdash \text{Con}(PA)$.
(b) If $R$ is an axiomatizable theory which is stronger than $S_2^1$, then $R \nvdash \text{Con}(R)$.

Proof: First note (b) implies (a). Let $\alpha$ be a unary function $\Sigma_1^1$-definable in $S_2^1$ such that the range of $\alpha$ is equal to the set of Gödel numbers of theorems of the $R$. Then $S_2^{i+1}$ 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 $PA_{A^4}$ denote the subsystem for Peano arithmetic $(PA)$ obtained by restricting induction to $\Sigma_1^1$ and $\Pi_1^1$-formulae. It is a classical result that $PA_{A^4+1} \vdash \text{Con}(PA)$. This can be proved by showing that $PA_4$ can formalize the proof of the cut elimination theorem and that $PA_{A^4+1}$ can define a truth valuation on $\Sigma_1^1$- and $\Pi_1^1$-formulae. Consequently, $PA_{A^4+1} \vdash \text{Con}(PA_{A^4})$. From this, it follows immediately that $PA_{A^4+1}$ is strictly stronger than $PA_4$ since by the Gödel incompleteness theorem, $PA_4 \nvdash \text{Con}(PA_4)$.

One way we might prove that $S_2^i$ is not equivalent to $S_2^{i+1}$ would be to adapt the proof that $PA_4$ is not equivalent to $PA_{A^4+1}$. Now it is certainly false that $S_2^{i+1} \dashv \vdash \text{Cons}(S_2^i)$; indeed, $S_2^i \vdash \text{Con}(Q)$, 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 $S_2^{i+1} \vdash \text{BDCon}(S_2^i)$ or $S_2^{i+1} \vdash \text{FCFCon}(S_2^i)$. 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 $i \geq 1$, $S_2^{i+1} \vdash \text{BDCon}(S_2^i)$. The author does not know whether $S_2^{i+1} \vdash \text{FCFCon}(S_2^i)$, but he conjectures that it is not the case.
Definition: In order to improve readability, we shall use the symbols $\mathcal{E}$ to denote "proves by bounded proof". This symbol will only be used metamathematically. For example, if $\Psi$ is a bounded formula,

$$S_1^{\mathcal{E}} \Psi$$

denotes the formula

$$(\exists w) Pr/BD_{A}(w, \llbracket \Psi \rrbracket)$$

which is a formula that asserts that there is a bounded $S_1^{\mathcal{E}}$-proof of $\Psi$.

If $\Psi$ is not a bounded formula, we can still sometimes define a formula $S_1^{\mathcal{E}} \Psi$. Namely, if $\Psi$ is $(\forall x)A(x)$, let $a$ be a new free variable. Then $S_1^{\mathcal{E}}(\forall x)A(a)$ is defined to be the formula

$$S_1^{\mathcal{E}} A(a)$$

where $a$ is a new free variable not appearing in $A$. If $\Psi$ is $(\exists y)A(y)$ and $A$ is a bounded formula then $S_1^{\mathcal{E}} \Psi$ is defined to be the formula

$$(\exists w)(\exists \beta)[Term(w)\land Pr/BD_{A}(w, \text{Sub}(\llbracket (\exists y)A \rrbracket, \llbracket \beta \rrbracket, w))]$$

where $\beta$ 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 $S_1^{\mathcal{E}} \Psi$ is the formula which asserts that there is a term $t$ and a bounded $S_1^{\mathcal{E}}$-proof $P$ such that $P$ is a proof of $(\exists y)(a)y)|A(a,y)$, where $a$ is a new free variable.

Proposition 14: Let $\Psi(a)$ be any bounded formula. Suppose $S_2 \vdash (\forall z)\Psi(z)$. Then

$$S_2 \vdash (\forall z)(S_1^{\mathcal{E}} \Psi(I_a)).$$

By our conventions for abbreviating formulae, the conclusion of Proposition 14 is an abbreviation for

$$S_2 \vdash (\forall z)(\exists w)Pr/BD_{(\forall z)\Psi(I_a)}(w, \text{Sub}(\llbracket \Psi \rrbracket, \llbracket \begin{array}{c} z \\ a \end{array}, x))).$$

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 $S_1$. We start with a bounded proof $P$ with consequent $\rightarrow \Psi(a)$. By Theorem 4.9, $P$ may be assumed to be restricted by parameter variables. $S_2$ 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, $S_2$ does not prove "Given a proof $P$ of $(\forall z)\Psi(z)$ and given a number $n$, there is an induction free proof of $\Psi(I_n)$." Instead, given a proof $P$ of $(\forall z)\Psi(z)$, $S_2$ proves "given a number $n$, there is an induction free proof of $\Psi(I_n)$."

Q.E.D. □

Definition: $S_2 + BDC_{\Pi^1_\text{f}}(S_2)$ is the theory $S_2$ plus the bounded axiom $\neg \Pr(BDC_{\Pi^1_\text{f}}(a, \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \right
Corollary 17: Either $S_2^1 \upharpoonright BDC(S_2^{1+3})$ or $S_2^1 \upharpoonright BDC(S_2^1)$. 

**Proof:** Suppose $S_2^1 \upharpoonright BDC(S_2^{1+3})$. Then $S_2^1 \vdash [S_2^1 \upharpoonright BDC(S_2^{1+3})]$, and thus

$$S_2^1 \vdash BDC(S_2^1) \rightarrow BDC(S_2^1 \upharpoonright BDC(S_2^{1+3})).$$

So by Corollary 16, $S_2 \upharpoonright BDC(S_2^1)$. □

Corollary 18: Let $j$ be the least number (if any) such that $S_j^2 \vdash BDC(S_j^2)$. Then

(a) $S_j^2 \upharpoonright BDC(S_j^2)$, for all $i < j$ and all $k$.

(b) $S_j^2 \upharpoonright BDC(S_j^2)$, 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 $S_i^2 \upharpoonright BDC(S_i^{1+3})$.

As we remarked at the outset of this section, these results were motivated by a desire to show that $S_2^1$ and $S_2^{1+3}$ are distinct theories. From this viewpoint, Corollary 19 is a negative result in that it states that the formula $BDC(S_2^1)$ can not be used to separate the theories $S_2^1$ and $S_2^{1+3}$.

There are weaker formulae we could attempt to use to separate $S_2^1$ and $S_2^{1+3}$. For example, it is an open question whether $S_2^{1+3}$ can prove $BQC(S_2^1)$ or $FCF(S_2^1)$. The author conjectures that neither of $BQC(S_2^{1+3})$ and $FCF(S_2^{1+3})$ 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

$$(\forall x \leq c)(\forall y \leq a) \neg \text{Prf}(y, x, \text{Sub}(b, c, 1, x, y)).$$

Note that $\phi$ is a $\Pi^1_5$-formula, hence $\phi$ represents a co-$NP$ predicate. It is not difficult to see that $\phi$ is co-$NP$ complete.

**Definition:** Suppose $NP = co-NP$. Let $\psi$ denote some fixed $\Sigma^1_5$-formula so that

$$\mathbb{N} \models \phi(a, b, c) \equiv \psi(a, b, c).$$

**Definition:** Suppose $NP = co-NP$. Let $\phi$ and $\psi$ be as above. Then $W$ is the theory with the same language as $S^1_2$ and all the axioms of $S^1_2$ 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. $\neg \text{Prf}(y, x, \text{Sub}(b, c, 1, x, y))$

Strictly speaking, $W$ depends on the choice of $\psi$ and a better notation for this theory might be $W_\psi$. However, we shall keep $\psi$ fixed and suppress the subscript.

**Definition:** Let $\vec{a}$ be a vector $a_0, \ldots, a_n$. Then $I_{\vec{a}}$ denotes the vector $I_{a_0}, \ldots, I_{a_n}$.

The next proposition formalizes the claim that $\phi$ 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 $A(\mathcal{F})$, there is a $\Sigma_1^b$-formula $A_1$ and a $\Pi_1^b$-formula $A_2$ such that

$$W \vdash (A(\mathcal{F}) \rightarrow A_1(\mathcal{F})) \wedge (A_2(\mathcal{F}) \rightarrow A_2(\mathcal{F})).$$

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 $\Sigma_1^b$-replacement axioms are theorems of $S_2^1$ and by Corollary 2.1, there is a formula $B$ which is $\Delta_1^b$ with respect to $S_2^1$ and a term $s(\mathcal{F})$ such that

$$S_2^1 \vdash A(\mathcal{F}) \rightarrow (\forall y \leq s(\mathcal{F}))B(\mathcal{F}, y).$$

By Corollary 7.6(b), there are terms $r_1(\mathcal{F}, b)$ and $r_2(\mathcal{F}, h)$ such that

$$S_2^1 \vdash B(\mathcal{F}, b) \rightarrow (\exists z \leq r_1)PrfBD^b(\mathcal{F}, B(1_q, I_1)),$$

and

$$S_2^1 \vdash \neg B(\mathcal{F}, h) \rightarrow (\exists z \leq r_2)PrfBD^b(\mathcal{F}, \neg B(1_q, I_1)).$$

Since $W$ has an axiom asserting $BDCond(S_2^1)$, we have

$$W \vdash B(\mathcal{F}, b) \rightarrow (\forall z \leq r_1)PrfBD^b(\mathcal{F}, \neg B(1_q, I_1)).$$

Let $t(\mathcal{F}, b)$ be the term $\sigma[r_1]$. Then

$$W \vdash A(\mathcal{F}) \rightarrow (\forall y \leq s(\mathcal{F}))((\forall z \leq t(\mathcal{F}, s(\mathcal{F}))) \rightarrow PrfBD^b(\mathcal{F}, \neg B(1_q, I_1))).$$

In other words,

$$W \vdash A(\mathcal{F}) \rightarrow \phi(t(\mathcal{F}, s(\mathcal{F})), \neg B(1_q, s(\mathcal{F}))).$$

Let $C(\mathcal{F})$ be the formula $\psi(t(\mathcal{F}, s(\mathcal{F})), \neg B(1_q, s(\mathcal{F})))$. Then $C \in \Sigma_1^b$ and $W \vdash A \rightarrow C$. This establishes (b) for the case $A \in \Pi_1^b$. 

A Proof-Theoretic Statement Equivalent to \( \text{NP} = \text{co-NP} \)

If \( A \in \Sigma_1^p \), apply the above construction to \( \neg A \) to find \( C \in \Sigma_1^k \) such that \( \forall^e A \leftrightarrow \neg C \). So (b) holds for \( A \in \Sigma_1^p \).

It is now easy to prove (b) for all bounded formulas \( A \) by induction on the quantifier complexity of \( A \).

Q.E.D. □

**Corollary 2:** Suppose \( \text{NP} = \text{co-NP} \) and let \( W \) be as above. Then for every bounded formula \( A \),

\[
W \vdash [A(\vec{v}) \supset W^{\text{NP}} A(I_\vec{v})].
\]

**Proof:** Let \( A_{\vec{v}} \) be as in Proposition 1. Then

\[
W \vdash [W^{\text{NP}} (A(\vec{v}) \supset A_{\vec{v}}(\vec{x}))].
\]

Also, since \( A_{\vec{v}} \in \Sigma_1^p \) and by Theorem 7,<br

\[
W \vdash [A_{\vec{v}}(\vec{x}) \supset W^{\text{NP}} A_{\vec{v}}(\vec{x})].
\]

Hence,

\[
W \vdash [A(\vec{v}) \supset W^{\text{NP}} A(I_\vec{v})].
\]

Q.E.D. □

**Proposition 3:** Suppose \( R \) is a consistent theory extending \( S^1_2 \). Let \( A(\vec{v}) \) be any bounded formula in the language of \( S^1_2 \). If \( R \vdash (\forall \vec{x}) A(\vec{x}) \) then \( N \models (\forall \vec{x}) A(\vec{x}) \).

**Proof:** Suppose \( R \vdash (\forall \vec{x}) A(\vec{x}) \) but \( N \models \neg A(\vec{v}) \) for some fixed vector of integers \( \vec{v} \). Then \( S^1_2 \vdash \neg A(\vec{v}) \) since \( \neg A(\vec{v}) \) is a closed, bounded, true formula. But since \( R \) is an extension of \( S^1_2 \), \( R \) must be inconsistent and we have arrived at a contradiction! □

**Corollary 4:** Suppose \( R \) is a consistent extension of \( S^1_2 \) and \( R \) is axiomatized by bounded formulae. Then every theorem of \( R \) is true for \( N \).

**Definition:** A bounded theory is if \( R \) is axiomatized by bounded formulae. The axioms of \( R \) may contain free variables.

So by Corollary 4, every bounded, consistent extension of \( S^1_2 \) 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 if whenever $A$ is a bounded formula and $R \vdash (\forall \bar{x})(\exists y)A(\bar{x}, y)$ there is a term $t(\bar{x})$ such that

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

and such that $t$ is a term in the language of Bounded Arithmetic.

Proposition 5: Let $R$ be a bounded extension of $S^1_2$. Then $R$ is of polynomial growth rate.

Proof: This is an immediate consequence of Parikh's theorem. $\Box$

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 $S^1_2$ such that $R$ is consistent and finitely axiomatized and such that for every bounded formula $A$,

$$R \vdash (\forall \bar{x})A(\bar{x}) \supset R \vdash [\exists t(\bar{x})A(\bar{x})].$$

(c) There is a consistent, axiomatizable extension $R$ of $S^1_2$ which is of polynomial growth rate such that for every $A \in \Pi^0_1$,

$$R \vdash (\forall \bar{x})A(\bar{x}) \supset R \vdash A(\bar{x}).$$

(d) There is a consistent extension $R$ of $S^1_2$ such that for some polynomial $p(n_1, n_2, n_3)$,

$$N \models [\phi(a, b, c) \supset (\exists z \leq 2^{p(a, b, c)}) \forall x \exists y \phi(x, y, z)].$$

Proof:

(a)$\Rightarrow$ (b): Let $R$ be the theory $W$ as in Corollary 2.

(b)$\Rightarrow$ (c): This is immediate from Proposition 5.

(c)$\Rightarrow$ (d): This is easily proved by noting that $\phi \in \Pi^0_1$, using the definition of polynomial growth rate and applying Proposition 3.

(d)$\Rightarrow$ (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 $N \models \phi(n_1, n_2, n_3)$. Hence
A Proof-Theoretic Statement Equivalent to $NP=co-NP$

\[ N \models (a \land b \land c) \iff (\exists x \leq 2^{2^{(n+1)j+1}}) \text{Pred}[x, \lceil a, \lceil I_a, I_a \rceil \rfloor]. \]

The righthand side of this equivalence is a $\Sigma^1_1$ formula and hence represents an $NP$ predicate. Thus $a \land b \land c$ is in $NP$.

Q.E.D. $\Box$

The importance of Theorem 8 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

\[ N \models [f(x)=b \iff (\exists y \leq 2^{p(|x|)}) \text{Pred}[f(x), \lceil I_a \rceil = I_a \rceil]]. \]

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 $Conf(x)$ which is defined as follows:

**Definition:** Let $R$ be any axiomatizable theory. Then $Conf(x)$ is defined to be the formula

\[ (\forall y \leq x)(\neg \text{Pred}[y, \lceil \rightarrow \rceil]). \]

If $R$ is furthermore a bounded theory, then $Conf_B(x)$ is defined to be

\[ (\forall y \leq x)(\neg \text{Pred}[y, \lceil \rightarrow \rceil]). \]

The question is whether there are "short" $R$-proofs of $Conf(x)$ or $Conf_B(x)$ for some bounded theory $R$. For example, if we could show that for all bounded, consistent, axiomatizable extensions $R$ of $S^2_F$ there is no term $t(x)$ such that

\[ N \models (\forall x)(\exists y \leq t) \text{Pred}[t(x), \lceil \text{Conf}(I_y) \rceil] \]

then we would have shown that $NP=co-NP$. Unfortunately, we have the following result:
Proposition 7: Let $R$ be any bounded, consistent, axiomatizable extension of $S^2$. Then there is a bounded, consistent, axiomatizable extension $Q$ of $R$ such that

$$Q\vdash (\forall x)[Q \models \text{Con}_Q(I_x)].$$

Proposition 7 soundly destroys any hope of proving $NP \neq co-NP$ with the formula $\text{Con}_Q$ since it is immediate that

$$Q\vdash (\forall x)[\text{Con}_Q(x) \Rightarrow (Q \models \text{Con}_Q(I_x))].$$

Proof: Let $Q_0, Q_1, Q_2, \ldots$ be the following theories:

(a) $Q_0$ is $R$
(b) $Q_i = Q_0 + \text{Con}(Q_i)$
(c) $Q_{i+1} = Q_i + \text{Con}(Q_i)$

Let $Q$ be the theory $\bigcup Q_i$.

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 $\text{Con}(Q_i)$ is a formula with Gödel number $G_i$ such that $2^i \leq G_i \leq 2^{i+1}$ for each $i$ and some constant $\delta$. For each $i \geq 0$, $S^2$ can metamathematically discuss $Q_i$ and $S^2$ can define formulae such as $\text{Con}(Q_i)$.

$S^2$ can also metamathematically define the theory $Q$ in a straightforward manner. In particular, there is a $\Delta^0_1$-predicate of $S^2$ which recognizes the axioms of $Q$.

Since each theory $Q, Q_0, Q_1, \ldots$ contains $R$, they each admit $\Sigma^0_1$-PIND inferences.

Now suppose we wish to find a $Q$-proof of $\text{Con}_Q(I_n)$ for some $n \in \mathbb{N}$. 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 $\text{Con}(Q_m)$ has Gödel number $G_m > n$. Hence, no axioms $\text{Con}(Q_m)$ 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 $Q_j$-proof. $S^2$ can formalize this argument and hence

$$S^2 \vdash \text{Con}(Q_{j_n}) \Rightarrow \text{Con}_Q(n).$$

But now $Q \supseteq S^2$ and $Q$ has $\text{Con}(Q_{j_n})$ as an axiom, so $Q \vdash \text{Con}_Q(n)$. The size of the $S^2$-proof of $\text{Con}(Q_{j_n}) \Rightarrow \text{Con}_Q(n)$ is proportional to the length $|n|$ of $n$ and the size $|G_{j_n}|$ of the axiom $\text{Con}(Q_{j_n})$ is $\leq 2^{j_n} \leq (1+|n|)^9$. Hence there is a polynomial, independent of $n$, such that the Gödel number of the $Q$-proof of $\text{Con}_Q(n)$ is less than $2^{|n|}$.\]
Furthermore, $S^1_3$ and hence $Q$ can formalize the reasoning of the above paragraph. Thus

$$S^1_3 \vdash (\forall x)[Q^P \text{Con}_q(I_x)].$$

So,

$$Q \vdash (\forall x)[Q^P \text{Con}_q(I_x)].$$

Q.E.D. ■

Regarding Proposition 7 it should be noted (see Pudlak [23]) that

$$S^1_3 \vdash (\forall x)[S^1_3 \vdash \text{Con}_q(I_x)].$$

However, the author doubts that

$$S^1_3 \vdash (\forall x)[S^1_3 \vdash \text{Con}_q(I_x)].$$

What Proposition 7 asserts is that for some bounded extension $R$ of $S^1_3$,

$$R \vdash (\forall x)[R^P \text{Con}_q(I_x)].$$

There are lower bounds known for the length of any $R$-proof of $\text{Con}_q(x)$. They were originally proved by H. Friedman [19] and later by Pudlak [23]. Their techniques can be extended to give a lower bound on the size of bounded $R$-proofs of $\text{Con}_q^R(r)$. Namely, we have:

**Proposition 8:** Let $R$ be a bounded, consistent extension of $S^1_3$. Then for any term $r$ there is a term $q$ of the language of Bounded Arithmetic such that for all $n \in \mathbb{N}$ there is a bounded $R$-proof of $\text{Con}_q^R(q(I_n))$ with Gödel number less than $r(n)$.

**Proof:** by the method of H. Friedman [19] and Pudlak [23]. ■

Unfortunately, the lower bound of Proposition 8 is not good enough to show that $\text{NP} \neq \text{co-NP}$ and by Proposition 7 there is no way it can be improved significantly.

Proposition 7 destroyed our hope of using $A(x) = \text{Con}_q(x)$ to prove $\text{NP} \neq \text{co-NP}$. So what else can we try? Well, one possibility is to pick $A(x)$ to be some $\text{co-NP}$ 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 \( A(\overline{a}, \overline{b}) \) in the language of \( R \), the following is an axiom of \( R' \):
   \[
   T(\overline{a} \overline{b}) \rightarrow (\forall \overline{a} A(\overline{a}, \overline{b})).
   \]
4. In addition, \( R' \) has the axiom
   \[
   \text{Th}_R((0\rightarrow \neg \text{arrow})*a) \supset T(a).
   \]

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 \( A(x) \models Con_{ZF}(x) \) to prove \( NP \neq \text{co-NP} \). In this case we do not have the difficulties that arose in Proposition 7; namely, it is not the case that

\[
R \models (\forall x)[R \models Con_{ZF}(I_x)].
\]

Indeed, it is not the case that

\[
R' \models (\forall x)[R \models Con_{ZF}(I_x)].
\]

This is because \( R' \models [(R \models Con_{ZF}(I_x)) \supset Con_{ZF}(x)] \) and by Gödel's second incompleteness theorem \( R' \models (\forall x) \neg Con_{ZF}(x) \).

This inspires us to make the following conjecture:

**Conjecture:** For any bounded, consistent, axiomatized extension \( R \) of \( S^1 \),

\[
R \models (\forall x) \neg Con_{ZF}(x) \supset R \models \neg Con_{ZF}(I_x).
\]
It should be difficult to prove this conjecture as an affirmative resolution of the conjecture would be a proof that $NP \neq co-NP$. 
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 than 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 Hooi [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 \( V_1^2 \) and \( U_1^2 \) such that a function \( f \) is \( \Sigma_1^1 \)-definable in \( U_1^2 \) iff \( f \) is computable by a polynomial space bounded Turing machine (i.e., \( f \in \text{PSPACE} \)); similarly, \( f \) is \( \Sigma_1^{1\ast} \)-definable in \( V_1^2 \) iff \( f \) is computable by an exponential time Turing machine (i.e., \( f \in \text{EXPTIME} \)).

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 \( \text{PSPACE} \) and \( \text{EXPTIME} \) are obtained.


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 \( i, j \in \mathbb{N} \), \( \phi^i \) is a free \( j \)-ary second order predicate symbol and \( \phi^i_j \) is a bound \( j \)-ary second order predicate symbol. We shall use \( \psi, \phi, \chi, \ldots \) and \( \phi^1, \phi^2, \phi^3, \ldots \) 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 \( i, j \in \mathbb{N} \), \( \lambda^i_j \) is a free second order \( j \)-ary function variable and \( \lambda^i_j \) is a bound second order \( j \)-ary function variable. We use \( \lambda^1, \lambda^2, \lambda^3, \ldots \) and \( \lambda^1, \mu^1, \mu^2, \lambda^3, \ldots \) 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 \( x \in \mathbb{N} \), \( |f(x)| \leq t(x) \).

Second order quantifiers are of the form \( (\forall \phi) \), \( (\exists \phi) \), \( (\forall \lambda^i) \) and \( (\exists \lambda^i) \). 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 \( \Sigma^{1,4}, \Pi^{1,4} \) 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. \( \Sigma^{0,4} = \Pi^{0,4} = \Delta^{0,4} \) 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. \( \Sigma^{1,4}_{+} \) is the set of formulae such that
   - (a) \( \Pi^{1,4}_{-} \subseteq \Sigma^{1,4}_{+} \)
   - (b) If \( A \) is in \( \Sigma^{1,4}_{+} \), so are \( (\forall x \leq t)A \), \( (\exists x \leq t)A \), \( (\forall \lambda^i)A \) and \( (\exists \lambda^i)A \).
   - (c) If \( A \) and \( B \) are in \( \Sigma^{1,4}_{+} \), so are \( AB \) and \( A \lor B \).
   - (d) If \( A \in \Sigma^{1,4}_{+} \) and \( B \in \Pi^{1,4}_{-} \), then \( \neg B \) and \( B \lor A \) are in \( \Sigma^{1,4}_{+} \).

3. \( \Pi^{1,4}_{+} \) is the set of formulae such that
   - (a) \( \Sigma^{1,4}_{-} \subseteq \Pi^{1,4}_{+} \)
   - (b) If \( A \) is in \( \Pi^{1,4}_{+} \), so are \( (\forall x \leq t)A \), \( (\exists x \leq t)A \), \( (\forall \lambda^i)A \) and \( (\exists \lambda^i)A \).
   - (c) If \( A \) and \( B \) are in \( \Pi^{1,4}_{+} \), so are \( AB \) and \( A \lor B \).
   - (d) If \( A \in \Pi^{1,4}_{+} \) and \( B \in \Sigma^{1,4}_{+} \), then \( \neg B \) and \( B \lor A \) are in \( \Pi^{1,4}_{+} \).
(1) \( \Sigma^{1,3} \) and \( \Pi^{1,3} \) are the smallest sets satisfying (1)-(3).

So \( \Sigma^{1,3} \) is the set of bounded first order formulae which may contain second order free variables but no second order quantifiers. \( \Sigma^{1,3} \) and \( \Pi^{1,3} \) 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 \( \Delta^{1,3} \), \( \Sigma^{1,3} \) and \( \Pi^{1,3} \) to be the subsets of \( \Delta^{1,3} \), \( \Sigma^{1,3} \) and \( \Pi^{1,3} \), 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):

\[
\frac{A(\alpha), \Gamma \rightarrow \Delta}{(\forall \phi)A(\phi), \Gamma \rightarrow \Delta}
\]

and

\[
\frac{A(\varphi'), \Gamma \rightarrow \Delta}{(\forall \lambda')A(\lambda'), \Gamma \rightarrow \Delta}
\]

(2) (second order \( \forall \)-right):

\[
\frac{\Gamma \rightarrow \Delta, A(\alpha)}{\Gamma \rightarrow \Delta, (\forall \phi)A(\phi)}
\]

and

\[
\frac{\Gamma \rightarrow \Delta, A(\varphi')}{\Gamma \rightarrow \Delta, (\forall \lambda')A(\lambda')}
\]

where \( \alpha \) and \( \varphi' \) are the eigenvariables of the inferences and must not appear in the lower sequent.

(3) (second order \( \exists \)-left):

\[
\frac{A(\phi), \Gamma \rightarrow \Delta}{(\exists \phi)A(\phi), \Gamma \rightarrow \Delta}
\]
and

\[
\frac{A(\phi') \Gamma \rightarrow \Delta}{(\exists \lambda') A(\lambda') \Gamma \rightarrow \Delta}
\]

where \(\alpha\) and \(\phi'\) are the eigenvariables of the inferences and must not appear in the lower sequent.

(4) (second order \(\exists\)right):

\[
\frac{\Gamma \rightarrow \Delta, A(\alpha)}{\Gamma \rightarrow \Delta, (\exists \phi) A(\phi)}
\]

and

\[
\frac{\Gamma \rightarrow \Delta, A(\phi')}{\Gamma \rightarrow \Delta, (\exists \lambda') A(\lambda')}
\]

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 \(\{y\} A(y)\) is a meta-expression, so "\(\{\,\}\)" and "\(\,\)" are not symbols in the syntax of second order logic.

The idea of an abstract is that \(\{y\} A(y)\) specifies a predicate which is true for those \(y\) such that \(A(y)\) holds. If \(F(\alpha)\) is a formula containing the free second order predicate variable \(\alpha\), we use \(F(\{y\} A(y))\) to denote the formula obtained by replacing every \(\alpha(\overline{y})\) in \(F\) by \(A(\overline{y})\). We will use metavariables \(V, U, \ldots\) to denote abstracts. The formal definition of what \(F(V)\) means is as follows:

Definition: If \(\alpha\) 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 \(\alpha\). \(F(V)\) is defined by induction on the complexity of \(F\):

1. If \(\alpha\) does not appear in \(F\) then \(F(V)\) is \(F\).
2. If \(F(\alpha)=\alpha(\overline{y})\), then \(F(V)\) is \(A(\overline{y})\).
3. If \(F\) is \(\neg B, B \land C, B \lor C\) or \(\exists C\). Then \(F(V)\) is \(\neg B(V), B(V) \land C(V), B(V) \lor C(V)\) or \(B(V) \exists C(V)\) respectively.
4. Suppose \(F(\alpha)\) is \((\forall z) B(\alpha)\) or \((\exists z) B(\alpha)\). If \(z\) appears in \(A\), we obtain \(A'\) by renaming the variable \(z\) in \(A\) to avoid conflict of variables. Then \(F(V)\) is \((\forall z) B(\{y\} A'(y))\) or \((\exists z) B(\{y\} A'(y))\) respectively.
5. Suppose \(F(\alpha)\) is \((\forall \phi) B(\alpha)\) or \((\exists \phi) B(\alpha)\). If \(\phi\) appears in \(A\), we obtain \(A'\) by renaming
the variable \( \phi \) in \( A \) to avoid conflict of variables. Then \( F(V) \) is \((\forall \phi) B(\{\{y\}A'(y)) \) or \((\exists \phi) B(\{\{y\}A'(y)) \), respectively.

**Proposition 1**: Let \( F \) be a formula and \( U \) and \( V \) be abstracts. Any second order theory of arithmetic proves the sequent

\[(\forall x)(U(x) \Rightarrow V(x)), F(U) \rightarrow F(V).\]

**Proof**: This is Proposition 15.13 of Takeuti [28] and is easily proved by induction on the complexity of the formula \( F \).

**Definition**: Let \( V = \{y\}A(y) \) 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 \( \Phi \) be a set of formulae. A \( \Phi \)-abstract is one of the form \( \{y\}A(y) \) where \( A \) is in \( \Phi \). \( \Phi \) is closed under substitution iff for every formula \( A \) in \( \Phi \) and every \( \Phi \)-abstract \( V \), \( A(V) \) is a formula in \( \Phi \).

We first define the comprehension axiom and rule for second order predicate symbols.

**Definition**: Let \( \Phi \) be a set of formulae closed under substitution. The \( \Phi \) comprehension axioms, \( \Phi-Ca \), are given by the axiom scheme:

\[(\forall x)(\forall \tilde{x} A(\{\{x\}\tilde{x}\}) \rightarrow A(\{y\}A(\{y\})) \rightarrow A(\{y\}A(\{y\}))\]

where \( A \) must be in \( \Phi \).

**Definition**: Let \( \Phi \) be a set of formulae closed under substitution. The \( \Phi \) comprehension rules, \( \Phi-CR \), are inferences of the forms:

1. \( (\Phi-CR; \exists \text{right}) \)

\[
\Gamma \rightarrow \Delta (\exists \phi) F(\phi)
\]

\[
\Gamma \rightarrow \Delta. F(V)
\]
(2) \( \Phi-CR; \Sigma; \lambda \leq \Delta \)

\[
F(V, \Gamma) \rightarrow \Delta \\
(\forall \phi) F(\phi, \Gamma) \rightarrow \Delta
\]

where \( \Sigma \) is both (1) and (2), \( V \) must be a \( \Phi \)-abstract. \( \Delta \) is called the principal abstract of the inference.

**Example:** Let \( F(\alpha) \) be the formula \((\exists y \leq a)(y \cdot y = a) \Rightarrow \alpha(a)\). Then if \( A \) is \((\exists y \leq a)(y \cdot y = a) \Rightarrow \alpha(a)\), \( F(A) \) is the formula

\[(\exists y \leq a)(y \cdot y = a) \Rightarrow (\exists y \leq a)(y \cdot y = a).\]

Since \( A \in \Sigma^{1, \lambda}_{\Phi} \), we can use \( \Sigma^{1, \lambda}_{\Phi} - CR \) to infer:

\[
\rightarrow (\forall z)(\exists y \leq x)(y \cdot y = x) \Rightarrow (\exists y \leq x)(y \cdot y = x) \\
\rightarrow (\exists \phi)(\forall z)(\exists y \leq x)(y \cdot y = x) \Rightarrow \phi(a)
\]

That is to say, \( \Sigma^{1, \lambda}_{\Phi} \)-comprehension implies that there is a predicate \( \phi \) which is true precisely for the perfect squares.

**Proposition 2:** Let \( \Phi \) be a set of formulae closed under substitution. Then the comprehension axioms \( \Phi-CA \) are equivalent to the comprehension rules \( \Phi-CR \).

**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. \( \Box \)

We next define the comprehension axiom and rules for function symbols.

**Definition:** Let \( \Phi \) be a set of formulae closed under substitution. The \( \Phi \)-function comprehension axioms, \( \Phi-FCA \), are given by the following axiom scheme:

\[
(\forall y)(\exists y')(\exists \lambda)(\forall \beta)(\forall \gamma)(\forall \delta)(\exists z \leq t)(A(z, y', \beta, \gamma, \delta)) \rightarrow (\exists y)( \forall \beta)(\forall \gamma)(\forall \delta)(A(z, y, \beta, \gamma, \delta))
\]

where \( A \) is any formula in \( \Phi \) and \( t \) is any term.

**Definition:** Let \( \Phi \) be a set of formulae closed under substitution. The \( \Phi \)-function comprehension rules, \( \Phi-FCR \), are inferences of the form:
Comprehension Axioms and Rules

(1) \( \Phi^\text{-FCR}, \exists\text{right} \)
\[
\frac{\Gamma \rightarrow \Delta, F(U)}{\Gamma \rightarrow \Delta, \exists \lambda \forall \gamma [P(V)]}
\]

(2) \( \Phi^\text{-FCR}, \forall\text{left} \)
\[
\frac{F(U), \Gamma \rightarrow \Delta}{(\forall \lambda \forall \gamma [P(V)], \Gamma \rightarrow \Delta}
\]

where for both (1) and (2), \( t \) is any term, \( U \) must be an abstract of the form \( (\forall \gamma) (\exists x \leq t) A(x, y) \) and \( V \) must be the abstract \( (\forall \gamma) A(\lambda(\forall \gamma), y) \), and \( A \) is required to be a formula in \( \Phi \). \( V \) is called the principal abstract of the inference.

**Proposition 3:** Let \( \Phi \) be a set of formulae closed under substitution. The \( \Phi^\text{-FCR} \) rules are equivalent to the \( \Phi^\text{-FCA} \) axioms.

**Proof:**

First we show that \( \Phi^\text{-FCR} \implies \Phi^\text{-FCA} \). Let \( A \in \Phi \). Using (\( \Phi^\text{-FCR}, \exists\text{right} \)) we can infer
\[
(\exists x \leq t) A(x, y) \rightarrow (\exists x \leq t) A(x, \delta)
\]
\[
(\forall \forall \forall [\exists x \leq t] A(x, y) \rightarrow (\exists x \leq t) A(x, y))
\]
\[
(\exists \lambda \forall \gamma [\exists x \leq t] A(\lambda(\forall \gamma), y) \rightarrow (\exists x \leq t) A(x, y))
\]

From this, the first and second order \( \forall \text{right} \) infernces give the \( \Phi^\text{-FCA} \) axiom for \( A \).

\( \iff \) The reverse implication is even easier.

Q.E.D. \( \square \)

9.3. Axiomatizations of Second Order Bounded Arithmetic.

The weakest second order theories of Bounded Arithmetic are obtained by enlarging the first order theories \( S_b^2 \) and \( T_b^2 \) to include second order variables.

**Definition:** We define a hierarchy, \( S_b^\lambda(\alpha, \beta) \) and \( \Pi_b^\lambda(\alpha, \beta) \) of the second order formulae which contain no second order quantifiers. The definition of \( S_b^\lambda(\alpha, \beta) \) and \( \Pi_b^\lambda(\alpha, \beta) \) is completely analogous to the definition of \( S_b^1 \) and \( \Pi_b^1 \) in §21.1, the only difference being that free second order variables may appear without restriction in the formulae. The sets \( S_b^\lambda(\alpha) \) and \( \Pi_b^\lambda(\alpha) \) contain those formulae of \( S_b^1(\alpha, \beta) \) and \( \Pi_b^1(\alpha, \beta) \), respectively, which have no second order function variables.
**Definition:** $S^2(\alpha, \xi)$ is the second order theory with second order function and predicate variables and the following axioms:

1. **Basic** axioms,
2. For each function variable $g^i$, the axiom $(\forall \xi)(g^i(\xi) \leq t(\xi))$,
3. The $\Sigma^1_1(\alpha, \xi)$-**PIND** axioms.

$S^2(\alpha)$ is the second order theory with only second order predicate variables (but no second order function variables). The axioms for $S^2(\alpha)$ are:

1. **Basic** axioms,
2. The $\Sigma^1_1(\alpha)$-**PIND** axioms.

$S^2(\alpha)$ is the union of the theories $S^2(\alpha)$ and $S^2(\alpha, \xi)$ is the union of the theories $S^2(\alpha, \xi)$. $T^2(\alpha, \xi)$, $T^2(\alpha)$, $T^2(\alpha)$, and $T^2(\alpha)$ are defined similarly using the **IND** axioms instead of the **PIND** axioms.

All of our earlier work on $S^2(\alpha)$ can be relativized to $S^2(\alpha, \xi)$. For example, the relativization of Theorem 2.6 holds and, for all $r \geq 1$, $S^2(\alpha, \xi)$ proves the $\Sigma^1_r(\alpha, \xi)$-**LIND** axioms. Another result which carries over is Theorem 2.7: by essentially the same proof as before we can show that $S^2(\alpha)$ can $\Sigma^1_r(\alpha)$-define the function

$$f(\alpha) = (\# z \leq |\alpha|)(\alpha(z)).$$

Also $S^2(\alpha, \xi)$ 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 $\eta^r_k$ of §5.4 were syntactically equivalent to second order function variables. Thus the theories $S^2(\alpha)$ and $S^2(\alpha, \xi)$ satisfy a relativized version of the Main Theorem 5.6.

**Definition:** $U^2_{\xi}$ 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 $S^2(\alpha, \xi)$,
2. $\Sigma^1_{\omega, \xi}$-comprehension axioms, ($\Sigma^1_{\omega, \xi}$-CA and $\Sigma^1_{\omega, \xi}$-FCA),
3. $\Sigma^1_{\omega, \xi}$-**PIND** axioms.

$U_{\xi}$ is the theory $\bigcup_{\xi} U^2_{\xi}$.

**Definition:** $\mathcal{U}_{\xi}$ is a second order theory of Bounded Arithmetic with predicate variables but no function variables. The axioms of $\mathcal{U}_{\xi}$ are
(1) All axioms of $S^3_2(\alpha)$.
(2) $\Sigma^b_4$-comprehension axioms ($\Sigma^b_4$-CA).
(3) $\Sigma^b_4$-PIND axioms.

$\bar{U}_2$ is the theory $\bigcup \bar{U}_n$.

**Definition:** $V^*_2$, $\bar{V}^*_2$, $V_2$ and $\bar{V}_2$ are defined exactly like $U^*_2$, $\bar{U}^*_2$, $U_2$ and $\bar{U}_2$ (respectively) except that IND axioms are used instead of PIND axioms.

**Proposition 4:** $(i \geq 1)$. $V^*_2 \vdash U^*_2$ and $\bar{V}^*_2 \vdash \bar{U}^*_2$.

**Proof:** $\Sigma^b_4$-IND $\Rightarrow$ $\Sigma^b_4$-LIND is trivial. $\Sigma^b_4$-LIND $\Rightarrow$ $\Sigma^b_4$-PIND is readily established by using the method of the proof of Theorem 2.11. These implications show that $V^*_2 \vdash U^*_2$. $V^*_2 \vdash U^*_2$ 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:** $U^*_2$ is a conservative extension of $\bar{U}^*_2$. $V^*_2$ is a conservative extension of $\bar{V}^*_2$.

Theorem 5 is proved by a series of lemmas. The most important one is Lemma 6.

**Lemma 6:** Let $A$ be a $\Sigma^b_4$-formula with no free second order function variables. Then there is a $\Sigma^b_4$-formula $A^*$ such that

$U^*_2 \vdash A \rightarrow A^*$.

**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 $(\forall \bar{y})G(\zeta, \alpha)$ asserts that the predicate $\alpha$ encodes the value of the function $\zeta$. When $\zeta$ is $k$-ary, $\alpha$ must be $(k+1)$-ary and we define $G(x, \alpha)$ to be the formula

$(\forall z < |\bar{y}|)[\alpha(x, \bar{z}, \bar{y}) \rightarrow \text{Bit}(x, \zeta(\bar{y})) = 1]$.

So $(\forall \bar{y})G(\zeta, \alpha)$ says that for all $x < |\bar{y}|$, $\alpha(x, \bar{z}, \bar{y})$ is true iff the $x$-th bit of the binary expression for $\zeta(\bar{y})$ is 1. Since $\zeta(\bar{y}) \leq \bar{y}$ for all $\bar{y}$, $\alpha$ does indeed code the values of $\zeta$. $G$ is a metaformula rather than a formula since the definition of $G(\zeta, \alpha)$ depends on the term $\zeta$ and the arity of $\zeta$.

Note that $G(x, \alpha)$ is a $\Sigma^b_4$-formula (in fact, $G(x, \alpha)$ is a $\Delta^b_4(\zeta, \zeta)$-formula.) The $\Sigma^b_4$-CA axioms prove
\[(\forall \lambda)(\exists \phi)(\forall x)(\forall y) \phi(x, y) \Rightarrow \text{Bot}(x, \lambda y \phi(x, y)) = 1),\]

hence, \(U^2_\lambda \vdash (\forall \lambda)(\exists \phi)(\forall y) \phi(x, y) \Rightarrow \text{Bot}(x, \lambda y \phi(x, y)) = 1\).

Conversely, since \(S^2_\lambda(\alpha, \beta) \subseteq U^2_\lambda\), we can introduce a new \(\Sigma^1_\lambda(\alpha, \beta)\)-defined function symbol \(f^2_{\phi}\) in \(U^2_\lambda\) satisfying

\[(\forall \phi)(\exists \lambda)(\forall y) [t(y) \Rightarrow (\forall z < t(y))(\text{Bot}(x, f^2_{\phi}(y)) = 1 \Rightarrow \phi(x, y))].\]

By the \(\Sigma^2_\lambda\)-FCA axioms, \(U^2_\lambda\) can prove

\[(\forall \phi)(\exists \lambda)(\forall y) [t(y) \Rightarrow (\exists z \leq t(y))(t(y) = \min(t(y), f^2_{\phi}(y)) \Rightarrow \lambda y \phi(x, y) = \min(t(y), f^2_{\phi}(y))].\]

Let \(H(\phi, \alpha)\) be the metaformula \(\phi(y) \Rightarrow \min(t(y), f^2_{\phi}(y))\). It is now immediate that

\[U^2_\lambda \vdash (\forall \phi)(\exists \lambda)(\forall y) H(\lambda y \phi)\]

and since \(U^2_\lambda \vdash G(\phi, \alpha) \Rightarrow H(\phi, \alpha)\),

\[U^2_\lambda \vdash (\forall \lambda)(\exists \phi)(\forall y) H(\lambda y \phi)\]

We are now ready to construct the desired formula \(A^\ast\) equivalent to \(A\). For every second order function variable \(\lambda f^2_{\phi}\) in \(A\) we use a new second order predicate variable \(\psi_{ij}\). We replace each \((\forall y)\) by \((\forall \psi_{ij})\) or \((\exists \psi_{ij})\), respectively. Let \(h_{ij}\) be the \(\Sigma^1_\lambda(\alpha)\)-defined function such that

\[h_{ij}(y) = \min(t(y), f^2_{\phi}(y)).\]

Wherever \(\lambda f^2_{\phi}\) appears in \(A\) we replace it by \(h_{ij}(\phi)\). After all these replacements have been carried out we have the formula \(A^\ast\). The \(\Sigma^1_\lambda(\alpha)\)-defined function symbols \(h_{ij}\) can be removed by replacing them by their defining formulae.

It is clear that \(A^\ast\) is a \(\Sigma^2_\lambda\)-formula and that \(U^2_\lambda \vdash A^\ast\Rightarrow A^\ast\).

Q.E.D. \(\square\)

**Definition:** \(\bar{U}^2_\lambda\) is the theory \(U^2_\lambda\) extended to include second order function variables and \(\Sigma^2_\lambda\)-comprehension. (However, \(\bar{U}^2_\lambda\) does not include all the \(\Sigma^2_\lambda\)-PIND axioms.)

\(\bar{\bar{U}}^2_\lambda\) is the theory \(\bar{U}^2_\lambda\) extended to include second order function variables and \(\Sigma^2_\lambda\)-comprehension (but not all the \(\Sigma^2_\lambda\)-IND axioms.)
Lemma 7:
(a) $\bar{U}_2$ is a conservative extension of $\bar{U}_2^t$.
(b) $\bar{V}_2$ is a conservative extension of $\bar{V}_2^t$.

Proof: By the proof of Lemma 6, for every formula $A$ there is a formula $A^*$ such that $U_2^t \vdash A \leftrightarrow A^*$ (even if $A$ is not bounded). Furthermore, if $A \in \Sigma^{1,\omega}_1$ then $A^* \in \Sigma^{1,\omega}_1$. We claim that for all formulae $A$, if $\bar{U}_2^t \vdash A$ then $\bar{V}_2^t \vdash A^*$. This will suffice to prove Lemma 7 since $A^*$ is equal to $A$ when $A$ contains no second order function variables.

The claim is proved by induction on the number of inferences in an $\bar{U}_2^t$-proof of $A$. The only nontrivial case to consider is the $\Sigma^{1,\omega}_1$-FCH comprehension rules. However, $\bar{U}_2^t$ can emulate $\Sigma^{1,\omega}_1$-FCH by using the $\Sigma^{1,\omega}_1$-CR comprehension rule. We leave the details to the reader. $\Box$

Lemma 8:
(a) The $\Sigma^{1,\omega}_1$-PIND axioms are theorems of $\bar{U}_2^t$.
(b) The $\Sigma^{1,\omega}_1$-IND axioms are theorems of $\bar{V}_2^t$.

Proof: (a) This is immediate from Lemma 6 and the fact that $\bar{U}_2^t$ has the $\Sigma^{1,\omega}_1$-PIND axioms.
(b) is proved by the same argument. $\Box$

Definition:
Let $R$ be a second order theory of Bounded Arithmetic and let $A$ be any formula. Then $A$ is $\Delta^{1,\omega}_1$ with respect to the theory $R$ if there are formulae $B \in \Sigma^{1,\omega}_1$ and $C \in \Pi^{1,\omega}_1$ 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 $\Delta^{1,\omega}_1$ to mean that $A$ is $\Delta^{1,\omega}_1$ with respect to $R$.

Definition: $\bar{U}_2$ is a second order theory of Bounded Arithmetic with second order predicate variables but no function variables. The axioms of $\bar{U}_2$ are

1. All axioms of $\bar{U}_2$
2. $\Delta^{1,\omega}_1$-comprehension axioms ($\Delta^{1,\omega}_1$-CA).
$\bar{O}_2$ is the theory $\bigcup_{i} \bar{O}_i$.

Definition: $\bar{Y}_1^0$ and $\bar{Y}_2$ are defined analogously to $\bar{O}_1$ and $\bar{O}_2$. So $\bar{Y}_1^0$ and $\bar{Y}_2$ are the theories $\bar{Y}_1^0$ and $\bar{Y}_2$ (respectively) plus the $\Delta^1_{\text{CA}}$ axioms.

It is an immediate consequence of Lemma 6 that second order function variables may be added to the syntax of $\bar{O}_1$ or $\bar{Y}_1^0$ to obtain a conservative extension. Of course when we add second order function variables we may also use the $\Delta_{\text{CA}}^1$ axioms and the $\Sigma^1_{\text{CA}}$-IND axioms. However, for our purposes in §10.5 and §10.6, it is more convenient to work with the theories $\bar{O}_1$ and $\bar{Y}_1^0$ without second order function variables.


We next prove that Gentzen's cut elimination theorem holds for $\bar{O}_2$ and $\bar{Y}_2$. We will show in §9.7 that $\bar{O}_2$ and $\bar{Y}_2$ also satisfy a version of Gentzen's cut elimination theorem.

Definition: Let $A(a_1, \ldots, a_k, \alpha_1, \ldots, \alpha_l)$ be a formula with all free variables as indicated. We say that $B$ is a substitution instance of $A$ if $B$ is $A(t_1, \ldots, t_k, \nu_1, \ldots, \nu_l)$ where each $t_i$ is an arbitrary term and each $\nu_i$ is a $\Sigma^1_{\text{CA}}$-abstract.

Lemma 2: ($i \geq 0$.)
(a) If $A$ is a $\Sigma^1_{\text{CA}}$ ($\Pi^1_{\text{CA}}$) formula then every substitution instance of $A$ is a $\Sigma^1_{\text{CA}}$ ($\Pi^1_{\text{CA}}$) formula.
(b) Suppose $P$ is a $\bar{O}_1$-proof (respectively, $\bar{Y}_1^0$-proof) of $\Gamma \rightarrow \Delta$ and that every principal formula of a free cut inference in $P$ is a first order formula. Then there is a proof cut free $\bar{O}_1$-proof (respectively, $\bar{Y}_1^0$-proof) $P'$ of $\Gamma \rightarrow \Delta$.
(c) Suppose $P$ is a proof cut free $\bar{O}_1$-proof (respectively, $\bar{Y}_1^0$-proof) of $\Gamma \rightarrow \Delta$ and that $\alpha$ is a free variable appearing in $\Gamma \rightarrow \Delta$. Further suppose $V$ is a $\Sigma^1_{\text{CA}}$-abstract. Let $\gamma(V)$ and $\Delta(V)$ denote the contexts obtained by substituting $V$ for every occurrence of $\alpha$ in the formula $\gamma$ and $\Delta$. Then $\Gamma(V) \rightarrow \Delta(V)$ has a proof cut free $\bar{O}_1$-proof (respectively, $\bar{Y}_1^0$-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 $\alpha$ in formulas in $P$. It is easy to see by examining the allowable inferences that every inference is $P(V)$ is a valid inference of $\bar{O}_1$.
(respectively, \( \bar{V}_2 \)). In particular, (a) guarantees that \( \Sigma^{1,2}_{\omega} \text{-PIND} \) or \( \Sigma^{1,2}_{\omega} \text{-IND} \), and \( \Sigma^{1,2}_{\omega} \text{-CR} \) inferences are still valid after the substitution of \( V \) for \( \alpha \).

However, \( P(V) \) may fail to be a proof in that there may be initial sequents of \( P(V) \) of the form

\[
s_1=t_1, \ldots, s_n=t_n, A(s_1, \ldots, s_n) \Rightarrow A(t_1, \ldots, t_n)
\]

where \( V(\bar{x})A(\bar{x}) \). 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 \( \Gamma(V) \Rightarrow \Delta(V) \).

\( Q \) is not necessarily free cut free, as \( Q \) may contain free cuts with principal formula \( A(\bar{x}) \). 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 \( \Gamma(V) \Rightarrow \Delta(V) \). \( \square \)

**Theorem 10:** (Cut Elimination Theorem). Let \( P \) be a \( \bar{U}_2 \)-proof or a \( \bar{V}_2 \)-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 \( \bar{U}_2 \)- (or \( \bar{V}_2 \)-) 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

\[
\frac{\Gamma \Rightarrow \Delta}{\Pi \Rightarrow A}
\]

\[
\frac{\Gamma, \Pi \Rightarrow \Delta}{\Gamma, \Pi^* \Rightarrow \Delta^*, A}
\]

where \( \Pi^* \) and \( \Delta^* \) are obtained from \( \Pi \) and \( \Delta \) by removing all occurrences of \( A \). A mix inference is free if all of the occurrences of \( A \) in \( \Delta \) and \( \Pi \) 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 \( \Pi \). The left rank of \( P \) is the maximum distance of a sequent containing a direct ancestor of an occurrence of \( A \) in the cedent.
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

\[
\text{ord}(P) = \omega^2 \text{level}(P) + \omega \text{grade}(P) + \text{rank}(P)
\]

where level\((P)\) and grade\((P)\) are the level and grade of the principal formula of the final mix of \( P \), and \( \omega \) is the first infinite ordinal.

Thus it suffices to show that if \( P \) is a proof with no free mix inferences except for the final inference of \( P \) and if Theorem 10 is satisfied for all proofs \( P' \) with ord\((P') < \text{ord}(P)\), then \( P \) satisfies Theorem 10. We modify the proof of Lemma 5.4 of Takeuti [28].

Case (1): \( \text{rank}(P) = 2 \).

Case (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

\[
\begin{align*}
\Gamma &\to \Delta, B(\alpha) & B(V), \Pi &\to \Lambda \\
\Gamma &\to \Delta, (\forall \phi) B(\phi) & (\forall \phi) B(\phi), \Pi &\to \Lambda \\
\Gamma, \Pi &\to \Delta, \Lambda
\end{align*}
\]

where \( V \) is a \( \Sigma^1_\text{P} \)-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 \( \Gamma \to \Delta, B(V) \) from the free mix free proof of \( \Gamma \to \Delta, B(\alpha) \). Thus we have a proof \( Q \) such that the only free mix in \( Q \) is its last inference:

\[
\begin{align*}
\Gamma &\to \Delta, B(V) & B(V), \Pi &\to \Lambda \\
\Gamma, \Pi^# &\to \Delta^#, \Lambda
\end{align*}
\]

where \( \Pi^# \) and \( \Delta^# \) are \( \Pi \) and \( \Delta \) minus all occurrences of \( B(V) \).

By the induction hypothesis, there is a free mix free proof \( Q^* \) of \( \Gamma, \Pi^# \to \Delta^#, \Lambda \) since ord\((Q) < \text{ord}(P)\). 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)-(2.1.5.ii): Similar to [28].
Case (2.1.3.iii): Suppose $A = (\forall \phi)B(\phi)$, $V$ is a $\Sigma^1_3$-abstract and the last inferences of $P$ are:

$$
\Gamma \rightarrow \Delta \quad \frac{B(V), \Pi \rightarrow \Lambda}{(\forall \phi)B(\phi), \Pi \rightarrow \Lambda} \quad \Gamma, \Pi^* \rightarrow \Delta^* \Lambda
$$

where now $\Delta$ and $\Pi$, but not $\Gamma$, may contain occurrences of $(\forall \phi)B(\phi)$. The contexts $\Pi^*$ and $\Delta^*$ are $\Pi$ and $\Delta$ minus all occurrences of $(\forall \phi)B(\phi)$. Modify the end of $P$ to obtain a proof $P_1$ which ends as

$$
\Gamma \rightarrow \Delta \quad \frac{B(V), \Pi \rightarrow \Lambda}{\Gamma, B(V), \Pi^* \rightarrow \Delta^* \Lambda}
$$

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 $P_1$ of the consequent of $P_1$. Now consider the proof which ends

$$
\Gamma \rightarrow \Delta \quad \frac{B(V), \Gamma, \Pi^* \rightarrow \Delta^* \Lambda}{(\forall \phi)B(\phi), \Gamma, \Pi^* \rightarrow \Delta^* \Lambda} \quad \Gamma, \Gamma, \Pi^* \rightarrow \Delta^* \Delta^* \Lambda
$$

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 $\Gamma, \Pi^* \rightarrow \Delta^* \Lambda$.

The rest of the cases are similar.

Q.E.D. □

9.5. $\Sigma^1_3$-Defined Functions and $\Delta^1_3$-Defined Predicates.

The second order theories of Bounded Arithmetic are in many respects analogous to the first order theories $S^1_2$ and $T^1_2$. 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 formulæ of induction axioms and comprehension axioms.
Definition: Let $R$ be a second order theory of Bounded Arithmetic. Suppose $A(x,y)$ is a \(\Sigma^0_1\)-formula with all free variables indicated and that

\[
\begin{align*}
(1) & \vdash (\forall x)(\exists y \leq t)A(x,y) \\
(2) & \vdash (\forall x)\langle y\rangle (\forall x)(A(x,y) \land A(x,z)) \Rightarrow \langle y\rangle = \langle z\rangle.
\end{align*}
\]

Then we say that $R$ can \(\Sigma^0_1\)-define the function $f$ such that $N = (\forall x)A(x,f(x))$.

The \(\Sigma^0_1\)-defined function symbols and the \(\Delta^0_1\)-defined predicate symbols play the same role in the second order theories of Bounded Arithmetic as the \(\Sigma^0_1\)-defined function symbols and the \(\Delta^0_1\)-defined predicate symbols do in the first order theories $S^2_1$ and $T^2_1$. In particular, the analogues of Theorems 2.2, 2.3 and 2.4 hold for second order Bounded Arithmetic.

Definition: Let $f$ and $\bar{f}$ be new function and predicate symbols. The sets $\Sigma^0_1(f,\bar{f})$ and $\Pi^0_1(f,\bar{f})$ are sets of bounded formulae in the language of second order Bounded Arithmetic plus the symbols $f$ and $\bar{f}$. 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 \(\Sigma^0_1\)-define each of the functions $f$ and $\bar{f}$ and \(\Delta^0_1\)-define each of the predicates $\bar{f}$. Let $R^*$ be the theory obtained from $R$ by adjoining the new symbols $f$ and $\bar{f}$ and their defining axioms. Then, if \(f \geq 0\) and $B$ is a \(\Sigma^0_1(f,\bar{f})\)- (or a \(\Pi^0_1(f,\bar{f})\))-formula, there is a formula $B^{*} \land \Sigma^0_1$ (or $\Pi^0_1$, respectively) such that $R^* \vdash B^* \rightarrow B$.

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 $f$ be a vector of defined function symbols of $R$ and $\bar{f}$ be a vector of defined predicate symbols. Then $R(f,\bar{f})$ is the conservative extension of $R$ obtained by enlarging the language to include $f$ and $\bar{f}$ and including the defining axioms for these symbols.

Corollary 12: (i = 1):

(a) Let $f$ be a vector of \(\Sigma^0_1\)-defined function symbols of $U^2_1$ (respectively, $V^2_1$) and let $\bar{f}$ be a vector of \(\Delta^0_1\)-defined predicate symbols of $U^2_1$ (respectively, $V^2_1$). Then $U^2_1(f,\bar{f})$ (respectively, $V^2_1(f,\bar{f})$) has as theorems the \(\Sigma^0_1(f,\bar{f})\)-PIND axioms (respectively, the \(\Delta^0_1(f,\bar{f})\)-IND axioms).

(b) Let $f$ be a vector of \(\Sigma^0_1\)-defined function symbols of $\bar{U}^2_1$ (respectively, $\bar{V}^2_1$) and let $\bar{f}$ be a vector of \(\Delta^0_1\)-defined predicate symbols of $\bar{U}^2_1$ (respectively, $\bar{V}^2_1$). Then $\bar{U}^2_1(f,\bar{f})$ (respectively, $\bar{V}^2_1(f,\bar{f})$) has as theorems the \(\Sigma^0_1(f,\bar{f})\)-PIND axioms (respectively, the \(\Delta^0_1(f,\bar{f})\)-IND axioms).

(c) Let $f$ be a vector of \(\Sigma^0_1\)-defined function symbols of $\bar{U}^2_1$ (respectively, $\bar{V}^2_1$) and let $\bar{f}$ be a vector of \(\Delta^0_1\)-defined predicate symbols of $\bar{U}^2_1$ (respectively, $\bar{V}^2_1$). Then $\bar{U}^2_1(f,\bar{f})$
§0.5  $\Sigma_1^L$-Defined Functions and $\Delta_1^L$-Defined Predicates

(respectively, $\hat{3}_N^L(f,g)$) has as theorems the $\Sigma_1^L$-PIND ($f,g$) axioms (respectively, the $\Sigma_1^L$-IND ($f,g$) axioms) and the $\Delta_1^L$-CA axioms.

Corollary 12 tells us that $\Sigma_1^L$-defined function symbols and $\Delta_1^L$-defined predicate symbols may be used freely in the principal formulas of induction inferences. Furthermore, if we are working in the theory $\hat{3}_N^L$ or $\hat{3}_N^L$ we may use such function and predicate symbols freely in principal abstracts of comprehension inferences.

The next two theorems give an application of $\Delta_1^L$-comprehension to show that $\hat{3}_N^L$ and $\hat{3}_N^L$ can define the iteration of $\Delta_1^L$-defined predicates. It is an open question whether these theorems hold for the theories $\hat{3}_N^L$ and $\hat{3}_N^L$.

**Theorem 13:** Let $A(a,b,x)$ and $B(a,b,x,a_1,x)$ be $\Delta_1^L$-formula's of $\hat{3}_N^L$, where $a_1$ is a unary predicate variable. Let $h(b,x)$ be a term which contains only the free variables $b$ and $x$. Then the predicate $K(a,b,x)$ which satisfies

$$K(a,b,x) \iff \begin{cases} A(a,b,x) & \text{if } b=0 \text{ and } a \leq h(b,x) \\ 0-1 & \text{if } a > h(b,x) \\ B(a,b,x,h,b,x,a_1,x) & \text{otherwise} \end{cases}$$

is $\Delta_1^L$-definable by $\hat{3}_N^L$.

**Proof:** The idea, of course, is to define $K(a,b,x)$ by induction on the length of $b$. Let $B^*(a,b,x,a_1,x)$ be the formula

$$B^*(a,b,x,a_1,x) = \{a_1(x,b,x,h,b,x,a_1,x)\}$$

and let $D(u,\phi)$ be the formula

$$\{(\forall y \leq 2^b)(\forall x \leq h(y,x))[(\phi(x)) \iff \phi^y(y)] \wedge (A(x,y) \iff (\phi^y(x)))\}.$$

It is easy to see that

$$\hat{3}_N^L \vdash (\exists \phi)\: D(0,\phi)$$

and

$$\hat{3}_N^L \vdash (\exists \phi)\: D(1,u,\phi) \iff (\exists \phi)\: D(u,\phi).$$

Hence, by $\Sigma_1^L$-PIND, $\hat{3}_N^L \vdash (\forall x)\: (\exists \phi)\: D(x,\phi)$. It is also not difficult to use $\Sigma_1^L$-PIND to prove
that

$$
\hat{\Theta}_2^1 \vdash D(u,\phi) \land D(u,\psi) \land \forall x \leq t(b, \tau) \forall y \leq \tau \exists z (\psi(x, y) \rightarrow \phi(x, y)).
$$

Hence, $\hat{\Theta}_2^1$ can $\Delta^1_{1, k}$-define $K$ by

$$
K(a, b, \tau, \gamma) \iff a \leq t(b, \tau) \land \exists \psi(D(b, \phi) \land \phi(a, b))
$$

and by the provably equivalent

$$
K(a, b, \tau, \gamma) \iff (\forall \psi)[D(b, \phi) \land \phi(a, b)].
$$

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 $a \leq t(b, \tau)$; otherwise the formula $D(u, \phi)$ could not be bounded. Theorem 13 is false without this restriction.

A similar use of $\Delta_{1, k}$-comprehension can be made by $\hat{\Theta}_2^1$:

**Theorem 14:** Let $A(a, x, \gamma)$ and $B(a, b, x, \alpha, \gamma)$ be $\Delta^1_{1, k}$-formulae of $\hat{\Theta}_2^1$ where $\alpha$ is a unary predicate variable. Let $t(b, \tau)$ be a term with the free variables indicated. Then the predicate $K(a, b, x, \gamma)$ which satisfies

$$
K(a, b, x, \gamma) \iff \begin{cases} 
A(a, x, \gamma) & \text{if } b = 0 \text{ and } a \leq t(b, \tau) \\
0 = 1 & \text{if } a > t(b, \tau) \\
B(a, b, x, \alpha, \gamma) & \text{if } K(z, b - 1, x, \gamma) \end{cases}
$$

is $\Delta_{1, k}$-definable in $\hat{\Theta}_2^1$.

The $\Delta^1_{1, k}$-formulae are in some respects more akin to the $\Delta^0_{1, k}$-formulae than to the $\Sigma^1_{1, k}$- and $\Pi^1_{1, k}$-formulae. For example, we have the following theorem:

**Theorem 15:**

(a) The $\Delta^1_{1, k}$-IND axioms and the $\Delta^1_{1, k}$-MIN axioms are theorems of $\hat{\Theta}_2^1$ and $\hat{\Theta}_3^1$.

(b) The $\Delta^1_{1, k}$-IND axioms and the $\Delta^1_{1, k}$-MIN axioms are theorems of $\hat{\Theta}_2^1$ and $\hat{\Theta}_3^1$. 
§9.5 \[ \Sigma^1_{\infty}\text{-Defined Functions and } \Delta^1_{\infty}\text{-Defined Predicates} \]

**Proof:** It is obvious that the \( \Delta^1_{\infty}\text{-IND} \) axioms are theorems of \( V_2^1 \). The fact that the \( \Delta^1_{\infty}\text{-IND} \) axioms are theorems of \( U_2^1 \) is proved just like Theorem 2.22.

Now we claim that the \( \Delta^1_{\infty}\text{-MIN} \) axioms follow from the \( \Delta^1_{\infty}\text{-IND} \) axioms. Indeed, the minimization axiom for a \( \Delta^1_{\infty} \)-formula \( A \) can be proved by using induction on the \( \Delta^1_{\infty} \)-formula \( (\forall y \exists x)(\neg A(y)) \). This proves (b).

(b) is proved similarly.

Q.E.D. \( \square \)

9.6. \( \Sigma^1_{\infty} \)-Replacement.

The \( \Sigma^1_{\infty} \)-replacement axioms of second order Bounded Arithmetic are analogous to the \( \Sigma^1_{\infty} \)-replacement axioms of first order Bounded Arithmetic. The \( \Sigma^1_{\infty} \)-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 \( \Sigma^1_{\infty} \)-replacement axioms, we need first to define an analogue of the Gödel beta function which operates on predicates.

**Definition:** Let \( \alpha \) be a second order unary predicate variable. We write \( \beta(b, \alpha) \) as an abbreviation for the atomic abstract \( \{x \mid \beta(b, x)\} \).

The motivation behind this definition of \( \beta \) is that it can be used as a Gödel beta function operating on predicate variables. One simple application of \( \beta \) is as a pairing function. Thus, we can think of the predicate variable \( \alpha \) coding the two predicates \( \beta = \beta(1, \alpha) \) and \( \gamma = \beta(2, \alpha) \). Conversely, given two unary predicates variables \( \beta \) and \( \gamma \), the \( \Sigma^1_{\infty} \)-comprehension axioms guarantee the existence of a predicate \( \alpha \) 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 \( \beta = \beta(1, \alpha) \) and \( \gamma = \beta(2, \alpha) \).

**Definition:** We write \( <\beta, \gamma> \) to denote the predicate \( \alpha \) defined as above. More precisely, \( <\gamma_1, \gamma_2> \) is an abbreviation for the abstract:

\[
\{x \mid (\exists z \leq x)((x = <1, z> \land \gamma_1(z)) \lor (x = <2, z> \land \gamma_2(z)))\}.
\]
As Theorem 16 below shows, $\mathcal{B}$ can be used for more sophisticated purposes than just as a pairing function.

**Definition:** The $\Sigma^{1,1}_i$-replacement axioms are the formulæ of the form

$$(\forall x \leq t)(\exists \beta)(A(x, \beta) \equiv (\exists \beta)(\forall x \leq t)A(x, \beta(x+1, \beta)))$$

where $t$ is any term, $\phi$ is a unary predicate variable, and $A$ is any $\Sigma^{1,1}_i$-formula. Other first and second order free variables may appear in $A$ as parameters.

**Theorem 16:** Let $i \geq 1$. Then $\Sigma^{1,1}_i$-replacement axioms are the theorems of both $U^i$ and $V^i$.

**Proof:** Let $A(k, \alpha)$ be a $\Sigma^{1,1}_i$-formula. Since $V^i$ is a stronger theory than $U^i$ (by Proposition 4), it will suffice to show that $U^i \vdash$ the replacement axiom for $A$.

One direction is easy:

$$U^i \vdash (\exists \beta)(\forall x \leq t)A(x, \beta(x+1, \beta)) \Rightarrow (\forall x \leq t)(\exists \beta)A(x, \beta).$$

The other direction is more tricky. Let $D$ be the formula $(\forall x \leq t)(\exists \beta)A(x, \beta)$. Let $B(c)$ be the formula

$$(\forall y < 2^{\text{cf}(\alpha)})(\exists \beta)(\forall x \leq 2^{\text{min}[\beta](x)})(2^{\text{min}[\beta](x)} + x \leq t \Rightarrow (\exists \beta)(y = 2^{\text{min}[\beta](x)} + x, \beta(x+1, \beta))).$$

Then it is obvious that $U^i \vdash D \supset B(0)$. Also it is straightforward to prove that $U^i \vdash D \supset B(c+1)$ by use of the $\Sigma^{1,1}_i$-comprehension axioms.

Since $B$ is a $\Sigma^{1,1}_i$-formula, $U^i \vdash D \supset B(0)$ follows from $\Sigma^{1,1}_i$-LIND. Finally, it is clear that

$$U^i \vdash B(0) \supset (\exists \beta)(\forall x \leq t)A(x, \beta(x+1, \beta)).$$

Hence the theorem is proved.

Q.E.D. □

Two more meta-predicates which are useful when used in conjunction with $\mathcal{B}$ are $ARY_4$ and $DEARY_4$.

**Definition:** Let $\alpha$ be a second-order unary predicate variable. We write $ARY_4(\alpha)$ as an abbreviation for the abstract $(x_1, \ldots, x_k)\alpha(x_1, \ldots, x_k)$. Let $\gamma$ be a $k$-ary predicate variable. We write $DEARY_4(\gamma)$ for the abstract $(x)\gamma(1, x), \ldots, \gamma(k, x))$. 
Hence $ARY_\alpha(DEARY_\alpha(\gamma))$ is the same predicate as $\alpha$. However, $DEARY_\alpha(ARY_\alpha(\alpha))$ is not in general the same as $\alpha$.

As an example of how $ARY_\alpha$ and $DEARY_\alpha$ can be used, consider the following more general form of the $\Sigma^{1,1}_\alpha$-replacement:

$$(\forall x \leq t)(\exists \phi^1_\delta)A(x,\delta,\phi^1_\delta)(\exists \phi^1_\delta)(\forall x \leq t)A(x,ARY_\delta(B(z+1,\phi^1_\delta)))$$

where $\phi^1_\delta$ and $\phi^1_\delta$ are unary and $k$-ary, respectively. Of course this more general form of the $\Sigma^{1,1}_\alpha$-replacement axiom is a consequence of the less general form presented above.

**Corollary 17:** Let $i \geq 1$. If $A$ is a $\Sigma^{1,1}_\alpha$-formula then there is a formula $B$ of the form $(\exists \phi)\psi$ such that $C$ is a $\Pi^{1,1}_\alpha$-formula and such that $U^i_\psi$ and $V^i_\psi$ prove that $A$ is equivalent to $B$.

**Proof:** By Lemma 6 we may assume without loss of generality that $A$ is a $\Sigma^{1,1}_\alpha$-formula. Now we may use prenex operations and the $\Sigma^{1,1}_\alpha$-replacement axioms to transform $A$ into the provably equivalent form

$$(\exists \phi_1) \cdots (\exists \phi_n)D(\phi_1, \ldots, \phi_n)$$

with $D \in \Pi^{1,1}_\alpha$. The $n$ second order existential quantifiers may be combined by use of the $\mathcal{B}$ function, giving $B$ equal to

$$(\exists \phi)(\mathcal{B}(1, \phi), \ldots, \mathcal{B}(n, \phi)).$$

Q.E.D. $\Box$

### 9.7. Cut Elimination in the Presence of $\Delta^{1,1}_\alpha$.Comprehension.

In this section we investigate cut elimination theorems for $\mathcal{B}$ and $\mathcal{B}$. 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 $\mathcal{B}$ and $\mathcal{B}$ satisfy cut elimination.

One difficulty with proving the cut elimination theorem for $\mathcal{B}$ and $\mathcal{B}$ is that it is possible for $A(\alpha)$ to be a $\Sigma^{1,1}_\alpha$-formula and $U$ to be a $\Delta^{1,1}_\alpha$-abstract and yet $A(U)$ is not a $\Sigma^{1,1}_\alpha$-formula. Thus Lemma 9(c) is not readily provable for $\mathcal{B}$ and $\mathcal{B}$ when $V$ is a $\Delta^{1,1}_\alpha$-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 $\Delta^1_{<k}$-abstract it is quite likely that the level of $B(y)$ is not less than the level of $(\forall x)B(x)$. However, without decreasing the level of the mix inference we can not apply the induction hypothesis to the proof by induction on $\text{ord}(P)$.

To circumvent these difficulties we shall define below theories $\hat{\text{L}_2}(\delta)$ and $\hat{\text{P}_2}(\delta)$ by enlarging the languages of $\hat{\text{L}_2}$ and $\hat{\text{P}_2}$. It will turn out that the constructive proof of the cut elimination used above in §9.4 can be extended to these expanded theories $\hat{\text{L}_2}(\delta)$ and $\hat{\text{P}_2}(\delta)$.

**Definition:** A relational $\delta$ is a predicate which sets on integers and predicates. More precisely, a $k$-ary relational $\delta$ is a subset of $$\mathbb{N}^k \times \omega_1^k \times \cdots \times \omega_1^k$$ where $n \geq 0$ and each $k \geq 1$ and $\omega^k_1$ 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 $\delta$ is introduced by a $\Delta^1_{<k}$-definition in $R$ if the following hold:

1. $A(\bar{a}, \bar{b})$ is a $\Sigma^1_{<k}$-formula, $B(\bar{a}, \bar{b})$ is a $\Pi^1_{<k}$-formula and $\bar{a}$ and $\bar{b}$ indicate all of the free variables in $A$ and $B$.
2. $\vdash A(\bar{a}, \bar{b}) \rightarrow B(\bar{a}, \bar{b})$.
3. The defining equation for $\delta$ is $$\delta(\bar{a}, \bar{b}) \iff A(\bar{a}, \bar{b}).$$

We will say that $\delta$ is $\Delta^1_{<k}$-defined by $R$ if the above holds and we write $R_\delta$ to denote the theory $R$ enlarged to include the new symbol $\delta$ and its two defining equations:

(a) $\delta(\bar{a}, \bar{b}) \rightarrow A(\bar{a}, \bar{b})$
(b) $A(\bar{a}, \bar{b}) \rightarrow \delta(\bar{a}, \bar{b})$

These two defining equations are valid initial sequents of the natural deduction system for $R_\delta$.

**Definition:** The theory $\hat{\text{L}_2}(\delta)$ is the following natural deduction system:

1. The BASIC axioms are initial sequents of $\hat{\text{L}_2}(\delta)$. Also, logical axioms and equality axioms are valid initial sequents of $\hat{\text{L}_2}(\delta)$.
2. $\Sigma^1_{<k}(\delta)$ and $\Pi^1_{<k}(\delta)$ are the sets of formulæ of the language of $\hat{\text{L}_2}(\delta)$ defined in the usual way by counting alternations of bounded quantifiers, ignoring sharply bounded quantifiers.
(3) If $A \vdash \Sigma^1_4(\delta)$ and $B \vdash \Pi^1_4(\delta)$ and $R \vdash A(\delta, \alpha) \leftrightarrow B(\delta, \beta)$, then the relational $\delta$ defined by
$$
\delta(\delta, \alpha) \iff A(\delta, \alpha)
$$
is a symbol of the language of $\bar{\mathcal{P}}_4^4(\delta)$. The two defining equations for $\delta$ are initial sequents of the natural deduction system for $\bar{\mathcal{P}}_4^4(\delta)$.

(4) The $\Sigma^1_4(\delta)$-PIND inferences are valid inferences of $\bar{\mathcal{P}}_4^4(\delta)$

(5) The $\Sigma^1_4(\delta)$-CR comprehension inferences are valid inferences of $\bar{\mathcal{P}}_4^4(\delta)$.

$\bar{\mathcal{P}}_4^4(\delta)$ is the theory $\bigcup \bar{\mathcal{P}}_4^4(\delta)$.

**Definition:** $\bar{\mathcal{P}}_4^4(\delta)$ and $\bar{\mathcal{P}}_4^4(\delta)$ are defined similarly to $\bar{\mathcal{P}}_4^4(\delta)$ and $\bar{\mathcal{P}}_4^4(\delta)$ except using $\Sigma^1_4(\delta)$-IND instead of $\Sigma^1_4(\delta)$-PIND.

**Definition:** Let $R$ be one of the theories $\bar{\mathcal{P}}_4^4(\delta)$, $\bar{\mathcal{P}}_4^4(\delta)$, $\bar{\mathcal{P}}_4^4(\delta)$ or $\bar{\mathcal{P}}_4^4(\delta)$. A formula $A$ is $\Delta^1_4(\delta)$ with respect to $R$ iff there is a $\Sigma^1_4(\delta)$-formula $B$ and a $\Pi^1_4(\delta)$-formula $C$ such that $R \vdash A \leftrightarrow B$ and $R \vdash A \leftrightarrow C$.

So, in effect, $\bar{\mathcal{P}}_4^4(\delta)$ and $\bar{\mathcal{P}}_4^4(\delta)$ are the same as the theories $\bar{\mathcal{P}}_4^4(\delta)$ and $\bar{\mathcal{P}}_4^4(\delta)$ except that all the $\Delta^1_4(\delta)$-defined relations are included in the language and only $\Sigma^1_4(\delta)$-CR comprehension is allowed.

**Proposition 18:**
(a) $\bar{\mathcal{P}}_4^4(\delta)$ is a conservative extension of $\bar{\mathcal{P}}_4^4(\delta)$.
(b) $\bar{\mathcal{P}}_4^4(\delta)$ is a conservative extension of $\bar{\mathcal{P}}_4^4(\delta)$.

**Proof:**
(a) We begin by showing that $\bar{\mathcal{P}}_4^4(\delta)$ is an extension of $\bar{\mathcal{P}}_4^4(\delta)$. For this it suffices to show that $\Delta^1_4$-comprehension is a derived rule of $\bar{\mathcal{P}}_4^4(\delta)$. So suppose $A \in \Sigma^1_4$, $B \in \Pi^1_4$ and $\bar{\mathcal{P}}_4^4(\delta) \vdash A \leftrightarrow B$. Let $V$ be the abstract $(\exists \alpha) A(\delta, \alpha, \bar{x})$ where $\exists$, $\bar{x}$ and $\bar{\alpha}$ indicate all the free variables of $A$ and suppose that $\bar{\mathcal{P}}_4^4(\delta)$ proves the sequent
$$
\Gamma \vdash \Delta, F(V).
$$
Let $\delta$ be the relational of $\bar{\mathcal{P}}_4^4(\delta)$ which is $\Delta^1_4$-defined by
\[ \delta(\vec{x}, \vec{y}) \iff A(\vec{x}, \vec{y}) . \]

Let \( V_i \) be the abstract \( \{\delta(\vec{x}, \vec{y})\}. \) Then there is a \( \hat{\delta}_i(\delta) \)-proof which ends

\[
\frac{\Gamma \rightarrow \Delta, F(\vec{V}) \quad F(\vec{V}) \rightarrow F(V_i)}{\Gamma \rightarrow \Delta, F(V_i)} \quad \frac{\Gamma \rightarrow \Delta, F(V_i)}{\Gamma \rightarrow \Delta, \delta(\vec{x}, \vec{y})F(\phi)}
\]

since \( V_i \) is a \( \Sigma^b_1(\delta) \)-abstract (in fact, it is an atomic abstract).

Hence \( \hat{\delta}_i(\delta) \) is an extension of \( \hat{\delta}_i^* \). The fact that \( \hat{\delta}_i(\delta) \) is conservative over \( \hat{\delta}_i^* \) is proved just like Corollary 12(c).

(b) is proved similarly to (a).

Q.E.D. \( \square \)

Because of the way we have defined the languages of \( \hat{\delta}_i(\delta) \) and \( \hat{\delta}_i^*(\delta) \) there will exist formulae \( F(\delta) \) such that \( F(\vec{V}) \) is not defined for \( \vec{V} \) an arbitrary abstract. In particular, if \( F \) is \( \delta(\sigma) \) for some relational \( \delta \), then \( \delta(\vec{V}) \) is not a formula and \( F(\vec{V}) \) is not defined. Thus we only allow \( \Sigma^b_1(\delta) \)-CR comprehension so be applied in those cases of the form

\[
\frac{\Gamma \rightarrow \Delta, F(\vec{V}) \quad F(\vec{V}) \quad F(\phi)}{\Gamma \rightarrow \Delta, C(\vec{x}, \vec{y})F(\phi)}
\]

where \( F(\delta) \) is a formula such that \( F(\vec{V}) \) is defined. Of course, \( F(\vec{V}) \) is defined iff \( \delta \) is not an argument to any \( \Delta^b_1 \)-defined relational in \( F(\delta) \).

We shall also need the capability to substitute a \( \Delta^b_1 \)-abstract \( V \) for \( \delta \) in an arbitrary formula \( F(\delta) \). Accordingly, we make the following definition:

**Definition:** Let \( R \) be one of the theories \( \hat{\delta}(\delta) \) or \( \hat{\delta}^*(\delta) \). Let \( \alpha \) be an \( n \)-ary predicate variable, \( F(\delta) \) be a formula in the language of \( R \), and \( V \) be the abstract \( \{y_1, \ldots, y_n\}A(\vec{y}, \vec{z}, \vec{w}) \) where \( A \) is a \( \Sigma^b_1(\delta) \)-formula of \( R \). Then \( F(\vec{V}) \) is defined by induction on the complexity of \( F \):

1. If \( \alpha \) does not appear in \( F \), then \( F(\vec{V}) \) is \( F \).
2. If \( F(\alpha) \) is \( \alpha(\vec{y}) \), then \( F(\vec{V}) \) is \( A(\vec{x}, \vec{y}, \vec{z}) \).
3. If \( F(\alpha) \) is \( \delta_C(\vec{x}, \alpha, \vec{y}) \) where \( C \) is a \( \Delta^b_1(\delta) \)-formula of the theory \( R \) and \( \delta_C \) is the relational with defining axiom
   \[ \delta_C(\vec{x}, \alpha, \vec{y}) \iff C(\vec{x}, \alpha, \vec{y}) \]
   then \( F(\vec{V}) \) is \( \delta(\vec{y}) \) where \( \delta \) is the relational defined by
\[ \delta(\overline{z}, \overline{\gamma}) \iff C[\overline{V}][\overline{\gamma}], \]

Here \( C[\overline{V}] \) is the result of substituting \( \overline{V} \) for \( \overline{\alpha} \) in \( C \). Notice that since \( A \) is a \( \Sigma^L_1(\delta) \)-formula and \( C \) is \( \Delta^L_1(\delta) \) with respect to \( R \), so is \( C[\overline{V}] \).

(a) Suppose \( F \) is \( \rightarrow B, BAC, BvC \) or \( B \supset C \). Then \( F[\overline{V}] \) is \( \neg B[\overline{V}], B[\overline{V}] \supset C[\overline{V}], B[\overline{V}] \lor C[\overline{V}] \) or \( B[\overline{V}] \supset \) \( C[\overline{V}] \), respectively.

(b) Suppose \( F(\overline{a}) \) is \( (\forall x) \beta(\overline{a}) \) or \( (\exists x) \beta(\overline{a}) \). If \( x \) appears in \( A \), we obtain \( A' \) by renaming the variable \( x \) in \( A \) to avoid conflict of variables. Then \( F[\overline{V}] \) is \( (\forall x) \beta[\overline{y}]A'(\overline{y}) \) or \( (\exists x) \beta[\overline{y}]A'(\overline{y}) \), respectively.

(c) Suppose \( F(\overline{a}) \) is \( (\forall \phi) \beta(\overline{a}) \) or \( (\exists \phi) \beta(\overline{a}) \). Since \( A \) has no second order quantifiers, the bound variable \( \phi \) does not appear in \( A \). So \( F[\overline{V}] \) is \( (\forall \phi) \beta[\overline{y}]A'(\overline{y}) \) or \( (\exists \phi) \beta[\overline{y}]A'(\overline{y}) \), respectively.

**Definition.** Let \( A(a_1, \ldots, a_s, a_{s+1}, \ldots, a_m) \) be a formula where the \( a \)'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 \( A[V_1, \ldots, V_n(t_1, \ldots, a_m)] \) where each \( V_i \) is a \( \Sigma^L_0(\delta) \)-abstract and is substituted in for \( a_i \) and each \( t_k \) is a term substituted in for \( a_k \).

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 \( \overline{\beta}_k(\delta) \) and \( \overline{\beta}_k(\delta) \).

**Lemma 12:** Let \( \geq 0 \) and let \( R \) be one of the theories \( \overline{\beta}_1(\delta) \) or \( \overline{\beta}_1(\delta) \).

(a) If \( B \) is a \( \Sigma^L_1(\delta) \)-formula (respectively, a \( \Pi^L_1(\delta) \)-formula), then every substitution instance of \( B \) is a \( \Sigma^L_1(\delta) \)-formula (respectively, a \( \Pi^L_1(\delta) \)-formula).

(b) Suppose \( P \) is an \( R \)-proof of \( \Gamma \rightarrow \Delta \) 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 \( \Gamma \rightarrow \Delta \).

(c) Suppose \( P \) is a free cut free \( R \)-proof of \( \Gamma \rightarrow \Delta \) and \( \alpha \) is a free variable appearing in \( \Gamma \rightarrow \Delta \). Further suppose \( V \) is a \( \Sigma^L_1(\delta) \)-abstract. Let \( \Gamma[V] \) and \( \Delta[V] \) denote the cedents obtained by substituting \( V \) for every occurrence of \( \alpha \) in formulæ in \( \Gamma \) and \( \Delta \). Then \( \Gamma[V] \rightarrow \Delta[V] \) 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 \( \alpha \) in formulæ in \( P \). It is easy to see by examining the allowable inferences that every inference in \( P[V] \) is a valid inference of \( \overline{\beta}_k(\delta) \) (respectively, \( \overline{\beta}_k(\delta) \)). In particular, (a) guarantees that \( \Sigma^L_1(\delta)-\text{PIND} \) or \( \Sigma^L_1(\delta)-\text{IND} \) and
$\Sigma^1_{0}(\delta)-CR$ inferences are still valid after the substitution of $V$ for $a$.

If $P$ contains any initial sequents which are defining axioms for some relational $\delta$, say

$$\delta(a_1, \ldots, a_n) \rightarrow C(a_1, \ldots, a_n)$$

then in $P \mid V$ this initial sequent becomes

$$\delta^*(a_1, \ldots, a_n) \rightarrow C\mid V(a_1, \ldots, a_n)$$

where $\delta^*$ is $\delta \mid V$. This is a defining axiom for $\delta^*$ and hence is a valid initial sequent.

However, if $P \mid V$ may fail to be a proof in that there may be initial sequents of $P \mid V$ of the form

$$s_1 = t_1, \ldots, s_n = t_n, A(s_1, \ldots, s_n) \rightarrow A(t_1, \ldots, t_n)$$

where $A$ is not atomic. However, sequents of this form are easy to prove without free cuts. So we merely tack on to $P \mid V$ free cut free proofs of these initial sequents and then obtain a proof $Q$ of $\Gamma \mid V \rightarrow \Delta \mid V$.

$Q$ is not necessarily free cut free since $Q$ may contain free cuts with principal formulas of the form $A\mid V$ where $A$ is atomic. But each $B\mid V$ is first order and so by (b) there is a free cut free $R$-proof of $\Gamma \mid V \rightarrow \Delta \mid V$.

Theorem 10: Let $R$ be one of the theories $\text{RT}_{2}(\delta)$ or $\text{RT}_{2}(\delta^*\delta)$ 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 $\text{ord}(P)$ of $P$ as before and proceed by induction on the $\text{ord}(P)$. The only difference is that in Case (1.5 iii) we use Lemma 19(c) instead of Lemma 9(c).

Q.E.D.

Corollary 11: Let $R$ be one of the theories $\text{RT}_{2}(\delta)$ or $\text{RT}_{2}(\delta^*\delta)$ where $i \geq 1$. Suppose $R$ proves the sequent $\Gamma \rightarrow \Delta$ and that every formula in $\Gamma \Delta$ is a $\Sigma^1_{i}(\delta)$-formula or a $\Pi^1_{i}(\delta)$-formula.

Then there is an $R$-proof $P$ of $\Gamma \rightarrow \Delta$ such that every formula in $P$ is in $\Sigma^1_{i}(\delta)$ or in $\Pi^1_{i}(\delta)$.
Proof: By Theorem 20 there is a free cut free proof $P$ of $\Gamma \rightarrow \Delta$. 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 $\Sigma^1_2 \delta$-formula since only $\Sigma^1_2 \delta$-PIND (or $\Sigma^1_2 \delta$-IND) inferences are allowed. In the second case, we claim that $A$ is in $\Sigma^1_2 \delta$. This is because each initial sequent must either (a) be an equality or BASIC axiom and contain only atomic formulæ or (b) be a defining equation for a relational.

Now it is clear that every formula in $P$ must be in $\Sigma^1_2 \delta$ or $\Pi^1_2 \delta$ 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 $\Sigma^1_2 \delta$-CR comprehension inferences are allowed, any comprehension inference of the form

$$
\frac{\Gamma \rightarrow A(V)}{\Gamma \rightarrow A(\exists \phi)A(\langle \phi \rangle)}
$$

will have $A \in \Sigma^1_2 \delta$ if $(\exists \phi)A(\langle \phi \rangle) \in \Sigma^1_2 \delta$.

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 $\Sigma^b_1$-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 $\Sigma^b_1$-definable in $U^2_1$ (or in $\bar{V}^2_1$) iff $f$ is computable by a polynomial space bounded Turing machine, i.e., iff $f$ is in PSPACE. In addition, $f$ is $\Sigma^b_1$-definable in $V^2_1$ (or in $\bar{V}^2_1$) iff $f$ is computable by an exponential time bounded Turing machine, i.e., iff $f$ is in EXPTIME.

13.1. EXPTIME functions are $\Sigma^b_1$-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 $p(n)$ so that the runtime of $M_f$ on input $x$ is always less than $2^{p(n)}$.

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 I. Recall that $\omega^n$ is equal to the set of $n$-ary predicates on the natural numbers.

**Definition:** Let $\phi_1, \ldots, \phi_i$ be predicate variables of a second order theory of Bounded Arithmetic, where each $\phi_i$ is $k_i$-ary. Then EXPTIME$(\phi_1, \ldots, \phi_i)$ 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

$$\mathbb{N}^{k_f} \times \omega^{k_1} \times \cdots \times \omega^{k_i}.$$
There is an oracle Turing machine $M_f$ with $r$ oracles such that for $1 \leq i \leq r$, the $i$-th oracle is $k_i$-ary and such that for all $\Omega_1, \ldots, \Omega_r$ with $\Omega_i \in \Omega_i^1$,

$$f(\vec{x}, \Omega_1, \ldots, \Omega_r) = M_f(\vec{x}, \Omega_1, \ldots, \Omega_r)$$

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 $p(\vec{x})$, 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 on oracles about $\Omega_i(\vec{x})$ for $\vec{x} \leq 2^{p(\vec{x})}$.

We will also denote $\text{EXPTIME}(\phi_1, \ldots, \phi_r)$ by $\text{EXPTIME}(\omega_1^1, \ldots, \omega_r^1)$.

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 $\phi \in \text{EXPTIME}$ and $M(x, \phi) \in \text{EXPTIME}(\phi)$ then $M(x, \phi) \in \text{EXPTIME}$. Without condition (5) this would not necessarily be true.

**Theorem:** Let $f$ be a function of polynomial growth rate in $\text{EXPTIME}$. Then $f$ is $\Sigma_1^1$-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 than $2^{p(\vec{x})}$ for each input $\vec{x}$, where $p$ is a suitable polynomial. Let the alphabet of $M$ be $\Gamma$ where the cardinality $|\Gamma|$ of $\Gamma$ is at least 3, and suppose that the symbols "0", "1" and "1" are included in $\Gamma$. Let the states of $M$ be $q_0, \ldots, q_2$ with $q_0$ the initial state. We set $\delta$ be a new symbol not in $\Gamma$. We assign arbitrarily the Gödel number to the phase $q_i$ of the symbols in $\Gamma$ and to "1"; we denote these Gödel numbers by $[q_i], [\delta], [\text{1}]$, 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

$$[\vec{x}], [\gamma_1], \ldots, [\gamma_k], \ldots, [\gamma_{k_{i+1}}], \ldots, [\gamma_{k_{i+n}}]$$

where each $\gamma_i$ is in $\Gamma$ and $\gamma_i$ is the current state of $M$, the current tape head position is at $\gamma_{i+1}$, and the $\vec{x}$'s denote the immovable part of the tape. To encode ID's of $M$ in the theory $V_2^1$, we shall use a second order function symbol $\phi$ with values less than or equal to $n$.

Let $\text{Next}(a_1, a_2, a_3, a_4, b)$ 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 $a_4$ in the next ID of $M$. For example, $b$ must equal $a_4$ 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 $\text{Next}(a_1, a_2, a_3, a_4, b)$ is true.
iff $t = \overline{\mathbf{S}}$. It is easy to see that \(\text{Next}_M\) is $\Sigma^1_{\text{b}}$-definable in $V^2_1$ (in fact, $\text{Next}_M$ is easily seen to be $\Sigma^1_{\text{b}}$-definable in $S^2_1$.)

Define $r(x)$ to be equal to $2^{\text{ord}(x)+1}x+2$, 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 $\phi^r$ so that for all $j \leq r(x)2^{\text{ord}(x)}$, $\phi^r(j)$ is equal to the $(\text{Rem}(j, r(x))+1)$-th number in the $(\lceil j/r(x) \rceil+1)$-th ID of the run of $M$ on input $x$.

Accordingly, we define a predicate $\text{Init}_{\mathcal{M}}(\phi^r)$ as:

\[
\text{Init}_{\mathcal{M}}(\phi^r, x) \iff (\forall i < r(x))((i-1 \lor i = \lceil \frac{x}{2} \rceil) \land \\
\land (i = 0 \lor i = r(x)-1 \lor i = \lceil \frac{x}{2} \rceil) \land \\
\land (i > 1 \land i \leq 2^{\text{ord}(x)} \lor i = \lceil \frac{x}{2} \rceil) \land \\
\land (i > r(x) - \lceil \frac{x}{2} \rceil - 2 \land i < r(x) - 1 \land 1 = \text{Bit}(r(x) - i - 2, x) \lor (i = \lceil \frac{x}{2} \rceil - 1) \land \\
\land (i > r(x) - \lceil \frac{x}{2} \rceil - 2 \land i < r(x) - 1 \land 0 = \text{Bit}(r(x) - i - 2, x) \lor (i = \lceil \frac{x}{2} \rceil - 1)).
\]

Then $\text{Init}_{\mathcal{M}}(\phi^r, x)$ asserts that the values of $\phi^r$ for $i < r(x)$ code the ID

\[
\#q_0 b b \cdots b b q_{|x|-1} \cdots q_0 \#\]

where $q_i$ 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 $\text{Init}_{\mathcal{M}}$.)

We define $\text{Run}_M(\phi^r, x)$ to mean that $\phi^r$ codes $i$ steps of the running of $M(x)$:

\[
\text{Run}_M(\phi^r, x) \iff \text{Init}_{\mathcal{M}}(\phi^r, x) \land \\
\land (\forall j < i)(\forall k < r(x)-2)(\text{Next}_M(\phi^r, x) \land (j \neq r(x)+k+1), \land \\
\land (j \neq r(x)+k+2, j \neq r(x)+k+3, j \neq r(x)+k+1)), \land \\
\land (\forall j < i)(\phi^r(j, r(x)+k+1) = \overline{\mathbf{S}} \lor (j+1 \land r(x) = \lceil \frac{x}{2} \rceil)).
\]

It is easy to see, by use of $\Sigma^1_{\text{b}}$-$\text{FCA}$, that

\[
V^2_1 \vdash (\exists \lambda^r)\text{Run}_M(\lambda^r, 0, x) \land \\
V^2_1 \vdash (\exists \lambda^r)\text{Run}_M(\lambda^r, \overline{\mathbf{S}}, x) \implies (\exists \lambda^r)\text{Run}_M(\lambda^r, \overline{\mathbf{S}}, x).
\]

Then, by an application of $\Sigma^1_{\text{b}}$-$\text{IND}$,

\[
V^2_1 \vdash (\exists \lambda^r)\text{Run}_M(\lambda^r, \overline{\mathbf{S}}, 2^{\text{ord}(x)}, x).
\]
Furthermore, the uniqueness condition is also provable, so

\[ V^1_2 \vdash \text{Run}_M(t^0, i, x_1). \text{Run}_M(t^0, i, x_2). (\forall y < (i + 1) \cdot t(x))(\phi^0(y) = \phi^0(y)). \]

We can easily \( \Sigma_1^{1,b} \)-define the functional \( \text{Value}_M \) such that if \( \phi^0 \) satisfies \( \text{Run}_M(t^0, \alpha_{\text{red}(0)}, x) \), then \( \text{Value}_M(\phi^0, x) \) is equal to the output of \( M \) which is coded in the last ID coded by \( \phi^0 \). \( \text{Value}_M \) is in fact polynomial time (relative to a function oracle for \( \phi^0 \)) and can be \( \Sigma_1^{1,b} \)-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

\[ A_M(x, y) \iff (\exists \lambda)(\text{Run}_M(\lambda^0, 2(t(x)), x) \land y = \text{Value}_M(\lambda^0, x)). \]

Because \( \text{Value}_M \) is polynomial time, we can assume without loss of generality that there is a term \( t_M(x) \) such that \( V^1_2 \) proves that \( (\forall \lambda)(\text{Value}_M(\lambda^0, x) \leq t_M(x)) \). Then

\[ V^1_2 \vdash (\forall x)(\exists y \leq t_M(x)) A_M(x, y). \]

We can now \( \Sigma_1^{1,b} \)-define \( f \) with the defining axiom

\[ f(x) = y \iff A_M(x, y). \]

Q.E.D. \( \Box \)

10.2. PSPACE functions are \( \Sigma_1^{1,b} \)-definable in \( U^1_2 \).

**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 \( p(\#) \) so that the total number of tape squares used by \( M_f \) on input \( x \) is always less than \( p(|x|) \).

**Definition:** Let \( \phi_1, \ldots, \phi_n \) be predicate variables of a second order theory of Bounded Arithmetic where each \( \phi_i \) is \( k \)-ary. Then \( \text{PSPACE}(\phi_1, \ldots, \phi_n) \) is the uniform set of functionals \( f \) such that the following hold:

1. \( \vdash (\exists x, y) (f(x, y) = y) \)
2. \( \vdash (\forall x)(\exists y \leq t_M(x)) A_M(x, y) \)
3. \( \text{EXPTIME}(\phi_1, \ldots, \phi_n) \)
4. \( \text{PSPACE}(\phi_1, \ldots, \phi_n) \)

\( \text{PSPACE}(c_\alpha^1, \ldots, \omega_1^1) \) is another name for \( \text{PSPACE}(\phi_1, \ldots, \phi_n) \).
There is no condition (5') in the definition above since condition (4') implies the condition (5) of the definition of EXPTIME($\phi$).

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 $\Sigma^b_1$-definable in $S^b_2$. A more general concept is that of bounded counting: a function $f$ is defined by bounded counting from $A$ if $f(x) = \#(y < x)A(y)$. Clearly, if $A$ is a PSPACE predicate, then $f$ is a PSPACE function and thus bounded counting should be definable in $U^2_2$.

We shall use the following scheme to express bounded counting: $\theta$ will be a function variable satisfying

$$\theta(x,y) = \begin{cases} \theta(2x,y) + \theta(2x+1,y) & \text{if } x < 2^{|y|} \\ 1 & \text{if } A(x = 2^{|y|}) \text{ and } 2^{|y|} \leq x \leq 2^{|y|} + y \\ 3 & \text{otherwise} \end{cases}$$

Then $\theta(1,y)$ is equal to the number of $x \leq y$ such that $A(x)$ holds.

**Proposition 2:** Let $A$ be a $\Sigma^b_1$-formula and let $t(x)$ be any term. Then the function

$$f(y) = (\#x \leq t(y))A(y,x)$$

is $\Sigma^b_{1,1}$-definable in $U^2_2$.

**Proof:** First we define $RDEF(t,x,y)$ to be the formula asserting that $(t,x,y)$ satisfies a condition similar to the definition of $\theta$ above, namely, $RDEF(t,x,y)$ is

$$\begin{align*}
&(x < 2^{|y|}) \land \theta(x,y) = \theta(2x,y) + \theta(2x+1,y) \\
&\land (x \geq 2^{|y|}) \land (x < 2^{|y|} + 1) \\
&\land (x \geq 2^{|y|} + 1) \land (\theta(x,y) \leq 1 \land (A(x = 2^{|y|}) \land \theta(x,y) = 1)) \\
&\land (x > 2^{|y|} + t) \land (\theta(x,y) = 1)
\end{align*}$$

Define $B(i,x)$ to be the formula

$$\exists z(\forall x \leq 2^{|t| + 1}) \|z \geq (1 + |i| - t) \land RDEF(z,x,y)\|.$$

An easy application of $\Sigma^b_{1,1}$-FCR shows that $U^2_1 \vdash B(0,x)$. Similarly, $U^2_1 \vdash B(i,x) \vdash B(Si,x)$. By $\Sigma^b_{1,1}$-LIND,

$$U^2_1 \vdash B(\|t + 1|,x).$$

Thus,
§10.2

PSPACE functions are $\Sigma^b_2$-definable in $U_2^1$

$$S_2^1 = \{ (y) | \exists z \leq 2^{|y|} \exists x (x < 2^{|y|} (1, y), (y, z) \in RDEF(y, x, y) \}.$$ 

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 $2^{|y|}$ 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 (1), the problem is divided into the two subproblems of computing $f(2)$ and $g(3)$. These subproblems are further divided into subproblems, etc. Thus to find $(\# y \leq t) A(y)$ we first find $(\# y < 2^{(\log t)}) A(y)$ and $(\# y \leq t - 2^{(\log t)}) A(y + 2^{(\log t)})$ 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

$$f^*(x, y) = \begin{cases} 0 & \text{if } |y| > p(|x|), y = 6 \\ g(x, y) & \text{if } |y| = p(|x|) \\ h(x, y, f^*(x, 2y), f^*(x, 2y + 1)) & \text{otherwise} \end{cases}$$

Then, for all $y$ and $x$, we must have $|f^*(x, y)| \leq q(|x|)$ and $f$ must satisfy the defining equation

$$f(x) = f^*(x, 1).$$

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 $\Sigma^b_2$-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 $\Sigma^b_2$-definable in $U_2^1$.

Proof: The proof is similar to the proof of Proposition 2. We first define $RDEF(x, y, z)$ to be the formula
\[ |y| > p([x]) \land |x| > y > 0 \land |y| = p([x]) \land |y| = p([x, y]) = \min (p(x, y), 2^{|y|^2})). \]

\[ |y| > p([x, y]) > |y| = p([x, y]) = \min (4(x, y, |x, y|, |x, y| + 1)2^{|y|^2}). \]

So \((\forall y < 2^{|y|^2}) RDEL_{c, y}(x, y)\) asserts that the \(c\) function is equal to the \(f^*\) function of the definition of limited recursion. Note that the \(\min\) function is used in the definition of \(RDEL_{c, y}\) to explicitly prevent the possibility of an overflow; that is to say, the possibility that some value of \(f^*(x, y)\) is too large.

We used \(g\) and \(h\) as function variables in the definition of \(RDEL_{c, y}\); since \(g\) and \(h\) are \(\Sigma^{0, 1}_{c}\)-definable, \(RDEL_{c, y}\) is a \(\Sigma^{0, 1}_{c}\)-formula. Let \(s([x])\) be the term \(2^{|x|^2}\) and define \(B(i, x, y)\) to be the formula

\[ (\forall y < 2^{|x|^2}) |y| > p([x]) = i \rightarrow RDEL_{c, y}(x, y). \]

It is easy to see using \(\Sigma^{0, 1}_{c}\)-comprehension that

\[ U^2_1 \vdash (\exists x') B(0, x, x') \]

and

\[ U^2_1 \vdash (\exists x') B(1, x, x'). \]

So by \(\Sigma^{0, 1}_{c}-\text{IND}\), \(U^2_1 \vdash (\exists x') B([p([x])], x, x'). \)

We also need to show that \(U^2_2\) proves that the \(x'\) is unique; that is, we need to show that

\[ U^2_2 \vdash B([p([x])], x, x') \land B([p([x])], x, x') \rightarrow (\forall y < 2^{|x|^2})(x', y) = c'(x, y)). \]

For this purpose, let \(C(i, x, x', y')\) be the formula

\[ (\forall y < 2^{|x|^2}) |y| > p([x]) = i \rightarrow x' = c'(x, y)) \]

and let \(D(i, x, x', y')\) be the formula

\[ B([p([x])], x, x') \land B([p([x])], x, x') \rightarrow c'(x, y)). \]

Then it is clear that

\[ U^2_1 \vdash D(i, x, x', y') \rightarrow C(i, x, x', y')) \]

and

\[ U^2_1 \vdash D(i, x, x', y') \rightarrow C(i, x, x', y')) \]

from which the desired uniqueness condition is obtained by an application of \(\Sigma^{0, 1}_{c}-\text{IND}\).
Let $A(x, z)$ be the formula

$$(\exists \lambda^x)[B(p(x), x, \lambda^x) \land \lambda^x = \lambda^y(x, 1)].$$

So $U_2^j \vdash (\exists \lambda^y)(\exists y \leq s(x))A(x, y)$. Also, for all $y$, $A(y, f(y))$ is true. Since $A$ is a $\Sigma_1^{1, k}$-formula, $f$ is by definition $\Sigma_1^{1, k}$-definable.

Q.E.D. □

We are now ready to prove that all PSPACE functions can be $\Sigma_1^{1, k}$-defined in $U_2^j$.

**Theorem 4:** Let $f$ be a function with polynomial growth rate in PSPACE. Then $f$ is $\Sigma_1^{1, k}$-definable in $U_2^j$.

**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 $f \in \text{PSPACE}$ if 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 $\Sigma_1^{1, k}$-definable in $U_2^j$. Theorem 3 thus implies that every PSPACE function of polynomial growth rate can be $\Sigma_1^{1, k}$-defined in $U_2^j$.

Q.E.D. □

### 10.3. Deterministic PSPACE Turing machines.

Theorem 4 established that $U_2^j$ can $\Sigma_1^{1, k}$-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 $U_2^j$ 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 $r(x) = |f(x)|$ with $r(x) \leq |x|/3$ for all $x$ so that $M$ is constrained by tape markers to run in space $r(x)$ on input $x$. Let $\text{Run}_M$ be defined exactly as in §10.1. Then our question is whether

$$U_2^j \vdash (\exists \lambda^x)\text{Run}_M(\lambda^x, 2^f(|x|), x)$$

where $f$ 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

$$U^2_{\lambda^*} \vdash (\forall y)(\exists x)^* \text{Run}_M(x,y,a).$$

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:

$$\text{STATE}_M(x,i,a) \equiv b \iff |b| \leq |x| \land t(a) \land \forall v \leq t(a) \exists n \in \text{MSP}(b,v,n,n) \land \phi(a,r(a)+v).$$

(Recall that $n$ bounds the Gödel numbers of symbols used to code states.) Thus STATE$_M(x,i,a)$ is equal to a number which codes the $i$-th state of the run coded by $\phi$. Let PRun$_M$ be the formula

$$\text{PRun}_M(c,i,a) \iff (\forall j \leq i)(\forall k \leq i(j \land k \land j(k+1),$$

$$\delta(j,r(a)+k+2), \delta(j,r(a)+k+3), j(j+1), r(a)+k+1)) \land$$

$$\forall \leq i \exists j : j(j+1) \land r(a)+1 \land \phi(j).$$

So PRun$_M(c,i,a)$ asserts that $c$ codes $i+1$ states of a run of $M$ except that no conditions are put on the initial state coded by $\phi$. Compare the definition of PRun$_M$ with the definition of Run$_M$.

Let D$_M(c,a)$ be the formula

$$D_M(c,a) \iff (\forall x \leq |x|)(\forall n \leq |n|)(\text{MSP}(x,n) \land \phi(x,a) \land \text{STATE}_M(x,a)).$$

D$_M(c,a)$ asserts that for all possible initial states there is a $\phi$ 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

$$U^2_{\lambda^*} \vdash D_M(0,a).$$

Also, we claim that

$$U^2_{\lambda^*} \vdash D_M(\lambda^*,a) \supseteq D_M(c,a).$$

To prove the claim we argue informally in $U^2_{\lambda^*}$. Suppose $D_M(\lambda^*,a)$ is true and that $x \leq |x| \land \phi(x,a)$ codes a state for $M$. Then there exists a $\lambda^*_x$ such that $x=\text{STATE}_M(\lambda^*_x,0,a)$ and such that PRun$_M(\lambda^*_x,\phi(x),a)$. So let $x \leq \text{STATE}_M(\lambda^*_x,\phi(x),a)$ and let $\lambda^*_x$ be such that
Since $D_y$ is a $\Sigma^b_1$-formula, we can use $\Sigma^b_1$-$PIND$ to deduce that

$$U^1_2 \vdash (\forall y)D_y(y,a).$$

From this Theorem 5 follows easily.

Q.E.D. □

10.4. Witnessing a $\Sigma^b_1$-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 $D^b_2(6)$ and $D^b_3(6)$.

We define Witness$_A$ below analogously to the way Witness was defined in §5.1. When $A$ is a $\Sigma^b_1(\delta)$-formula with free first order variables $\vec{d}$ and with free second order variables $\vec{a}$, we define Witness$_A^{\vec{d},\vec{a}}(\gamma,\vec{d},\vec{a})$ to be a $\Sigma^b_1(\delta)$-formula which asserts that $\gamma$ is a predicate which "witnesses" the truth of $A(\vec{d},\vec{a})$.

Although Witness$_A$ could readily be defined for arbitrary bounded formulae $A$, we shall forsake the added generality and restrict $A$ to be a $\Sigma^b_1(\delta)$-formula.

**Definition:** Suppose $A$ is a $\Sigma^b_1(\delta)$-formula. Let the free first order variables of $A$ be among $\vec{d}$ and the free second order (predicate) variables of $A$ be among $\vec{a}$. The $\Sigma^b_1(\delta)$-formula Witness$_A^{\vec{d},\vec{a}}(\gamma,\vec{d},\vec{a})$ is defined below, where $\gamma$ is a unary predicate variable. The definition is by induction on the complexity of $A$.

1. If $A$ is a $\Sigma^b_1(\delta)$-formula, then define

   $$\text{Witness}_A^{\vec{d},\vec{a}}(\gamma,\vec{d},\vec{a}) \iff A(\vec{d},\vec{a})$$

2. If $A$ is $B\lor C$, define

   $$\text{Witness}_A^{\vec{d},\vec{a}}(\gamma,\vec{d},\vec{a}) \iff \text{Witness}_B^{\vec{d},\vec{a}}(\gamma,\vec{d},\vec{a}) \land \text{Witness}_C^{\vec{d},\vec{a}}(\gamma,\vec{d},\vec{a})$$
(3) If $A \in B \cup C$, define
\[
\text{Witness}_{A}^{Z}(\gamma, \bar{a}, \bar{b}) \iff \text{Witness}_{B}^{Z}(\bar{b}(1, \gamma), \bar{b}, \bar{b}) \lor \text{Witness}_{C}^{Z}(\bar{b}(2, \gamma), \bar{b}, \bar{b})
\]

(4) If $A$ is $(\forall x \leq t)B(x)$, then define
\[
\text{Witness}_{A}^{Z}(\gamma, \bar{a}, \bar{b}) \iff (\forall x \leq t)\text{Witness}_{B}^{Z}(\bar{b}(x+1, \gamma), \bar{a}, \bar{b})
\]

(5) If $A$ is $(\exists x \leq t)B(x)$, then define
\[
\text{Witness}_{A}^{Z}(\gamma, \bar{a}, \bar{b}) \iff (\exists x \leq t)\text{Witness}_{B}^{Z}(\bar{b}(x, \gamma), \bar{a}, \bar{b})
\]

(6) If $A$ is $(\exists \phi)B(\phi)$ where $\phi$ is a $k$-ary predicate variable, then define
\[
\text{Witness}_{A}^{Z}(\gamma, \bar{a}, \bar{b}) \iff \text{Witness}_{B}^{Z}(\bar{b}(2, \gamma), \bar{a}, \bar{b}, \bar{b}, A \bar{R}(\bar{b}(1, \gamma)))
\]

(7) If $A$ is $\neg B$ and $A \in \Sigma^{1,4}_{0}(\delta)$, then define $\text{Witness}_{A}^{Z}$ by using prenex operations to transform $A$ so that it can be handled by cases (1)-(6). Specifically, if $A$ is $\neg (\neg B)$, $\neg (\neg B \land C)$, $\neg (\neg B \lor C)$, $\neg (\forall x \leq t)B$, $\neg (\exists x \leq t)B$ or $\neg (\exists \phi)B$; let $A^{*}$ be $B$, $\neg (\neg B \land C)$, $\neg (\neg B \lor C)$, $\neg (\forall x \leq t)\neg B$, $\neg (\exists x \leq t)\neg B$ or $\neg (\exists \phi)\neg B$. Then define
\[
\text{Witness}_{A}^{Z}(\gamma, \bar{a}, \bar{b}) \iff \text{Witness}_{A^{*}}^{Z}(\gamma, \bar{a}, \bar{b}).
\]

**Proposition 6:** Let $A(\bar{a}, \bar{b})$ be any $\Sigma^{1,4}_{0}(\delta)$-formula. Then $\overline{D}_{2}(\delta)$ and $\overline{P}_{2}(\delta)$ prove
\[
A(\bar{a}, \bar{b}) \iff (\exists \psi)\text{Witness}_{A}^{Z}(\psi, \bar{a}, \bar{b}).
\]

**Proof:** by induction on the complexity of $A$. The only nontrivial cases are (4) and (6) in the definition of $\text{Witness}_{A}^{Z}$.

**Case (4):** Suppose $A$ is $(\forall x \leq t)B(x)$. The induction hypothesis is that $\overline{D}_{2}(\delta)$ and $\overline{P}_{2}(\delta)$ prove
\[
B(\bar{a}, \bar{a}, \bar{b}) \iff (\exists \psi)\text{Witness}_{A}^{Z}(\psi, \bar{a}, \bar{a}, \bar{b}).
\]

By $\Sigma^{1,4}_{0}$-replacement (Theorem 9.16), $\overline{D}_{2}(\delta)$ and $\overline{P}_{2}(\delta)$ prove
\[
B(\bar{a}, \bar{a}, \bar{b}) \iff (\exists \psi)\text{Witness}_{A}^{Z}(\psi, \bar{a}, \bar{a}, \bar{b}).
\]
\[(\forall x \leq t)(\exists \phi) \text{Witness} \bar{\Sigma}^1_{\delta \bar{\delta}}(\phi; x, \bar{\delta}, \bar{\delta}) \quad \iff \quad (\exists \phi)(\forall x \leq t) \text{Witness} \bar{\Sigma}^1_{\delta \bar{\delta}}(\gamma; x, \bar{\delta}, \bar{\delta})\]

from which the desired result is immediate.

**Case (6):** Suppose \(A\) is \(\exists \delta \delta\) and that \(\hat{\delta}_1(\delta)\) and \(\hat{\delta}_2(\delta)\) prove

\[B(\xi, \beta, \phi) \iff (\exists \psi) \text{Witness} \bar{\Sigma}^2_{\delta \delta \delta}(\psi; \beta, \bar{\delta}, \phi)\]

where \(\phi\) is a \(k\)-ary predicate variable. From the definition of \(B\) and \(ARY_A\), we have immediately that \(\hat{\delta}_1(\delta)\) and \(\hat{\delta}_2(\delta)\) prove

\[(\exists \phi)(\exists \psi) \text{Witness} \bar{\Sigma}^2_{\delta \delta \delta}(\psi; \beta, \bar{\delta}, \phi) \iff \quad (\exists \phi)(\exists \psi) \text{Witness} \bar{\Sigma}^2_{\delta \delta \delta}(\beta; \psi, \bar{\delta}, ARY_A(\delta; 1, \psi))\]

and from this the desired result is immediate.

Q.E.D. \(\Box\)

As we remarked above, \(\text{Witness} \bar{\Sigma}^2_{\delta \delta}\) is a \(\Sigma^1_{\alpha \delta}(\delta)\)-formula whenever \(A\) is a \(\Sigma^1_{\alpha \delta}(\delta)\)-formula. The next proposition specifies the computational complexity of \(\text{Witness} \bar{\Sigma}^2_{\delta \delta}\).

**Proposition 7:** Let \(A(\bar{\delta}, \bar{\delta})\) be a \(\Sigma^1_{\alpha \delta}(\delta)\)-formula. Then \(\text{Witness} \bar{\Sigma}^2_{\delta \delta}(\gamma; \bar{\delta}, \bar{\delta})\) represents a predicate in \(\text{PSPACE}(\gamma; \bar{\delta})\).

**Proof:** This is an immediate consequence of the fact that \(\text{Witness} \bar{\Sigma}^2_{\delta \delta}\) contains no second order quantifiers. \(\Box\)

**Lemma 8:** Let \(A(\bar{\delta}, \bar{\delta}, \bar{\delta})\) be a \(\Sigma^1_{\alpha \delta}(\delta)\)-formula and let \(B(\gamma; \bar{\delta}, \bar{\delta})\) be a \(\Sigma^1_{\alpha \delta}(\delta)\)-formula, where the free variables of \(A\) and \(B\) are as indicated. Furthermore, \(\beta\) is a \(k\)-ary predicate variable and \(\bar{\theta}\) is a vector of \(k\) first order variables. Let \(U\) be the abstract \(\{\gamma\} B(\gamma; \bar{\delta}, \bar{\delta})\) and let \(A^*(\bar{\delta}, \bar{\delta})\) be the formula \(A(\bar{\delta}, \bar{\delta}, U)\). Then \(\hat{\delta}_3(\delta)\) and \(\hat{\delta}_4(\delta)\) prove

\[\text{Witness} \bar{\Sigma}^2_{\delta \delta}(\gamma; \bar{\delta}, \bar{\delta}, U) \iff \text{Witness} \bar{\Sigma}^2_{\alpha \delta \delta}(\gamma; \bar{\delta}, \bar{\delta}).\]

**Proof:** This is easily proved by induction on the complexity of \(A\). \(\Box\)

The final lemma of this section is not directly concerned with the \(\text{Witness} \bar{\Sigma}\) 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 \(\alpha\)'s
values but only on $\alpha$ restricted to some bounded domain.

**Lemma 5:** Let $A(\alpha, \beta, \gamma)$ be a bounded formula with all free variables as indicated. For notational simplicity, further suppose $\alpha$ is a unary predicate variable. Then there is a term $s_A(\delta)$ such that $\bar{B}_2(\delta)$ and $\bar{B}_2(\delta)$ prove

$$(\forall \gamma)(s_A(\delta)(\alpha(\gamma) \rightarrow B(\gamma)) \supset [A(\alpha, \beta, \gamma) \rightarrow A(\beta, \gamma, \gamma)].$$

**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 $\langle \cdots \rangle$ notation to apply to predicates. So

$$\langle \alpha_1, \ldots, \alpha_n \rangle$$

denotes $\langle \alpha_1, \ldots, \langle \alpha_{n-1}, \alpha_n \rangle \rangle$.

10.5. Only PSPACE is $\Sigma_1^{1b}$-definable in $U_1^b$.

In this section, the converse to Theorem 4 is proved. This establishes that a function $f$ of polynomial growth rate is $\Sigma_1^{1b}$-definable in $U_1^b$ if $f$ is computed by some polynomial space bounded (PSPACE) Turing machine. The main theorem of this section is:

**Theorem 10:** Suppose $A(\vec{x}, d)$ is a $\Sigma_1^{1b}(\delta)$-formula where $\vec{x}$ and $d$ are all the free variables of $A$. Also suppose $\bar{B}_2(\delta) \vdash (\forall \vec{z})(\exists y)A(\vec{x}, y)$. Then there is a $\Delta_1^{1b}(\delta)$-formula $B$, a term $t$ and a function $f$ so that

1. $\bar{B}_2(\delta) \vdash (\forall \vec{z})(\forall y)B(\vec{z}, y) \supset A(\vec{z}, y)$
2. $\bar{B}_2(\delta) \vdash (\forall \vec{z})(\exists y \leq t)B(\vec{z}, y)$
3. $\bar{B}_2(\delta) \vdash (\forall \vec{z})(\forall y)(\exists y)B(\vec{z}, y) \cdot B(\vec{x}, y) \supset y = z$
4. For all $n$, $N \models B(\vec{n}, f(\vec{n}))$
5. $f$ is a PSPACE function

Hence, $f$ is a PSPACE function which is $\Sigma_1^{1b}(\delta)$-definable in $\bar{B}_2(\delta)$ and for all $\mathcal{N}$, $A(\mathcal{N}, f(\mathcal{N}))$ is true.
The converse of Theorem 4 is an immediate corollary of Theorem 10:

**Corollary 11:** Suppose \( A(\mathcal{C},d) \) is a \( \Sigma^1_{1} \) formula where \( \mathcal{C} \) and \( d \) are all the free variables of \( A \). Also suppose \( U^{1}_{1} = (\forall \mathcal{F}(\exists y)A(\mathcal{F},y)) \). Then there is a PSPACE function \( f \) such that for all \( \alpha \), \( N = A(\alpha, f(\alpha)) \).

**Proof:** of Corollary 11 from Theorem 10:

By Lemma 9.6 and Theorem 9.5, we can assume without loss of generality that \( A \subseteq \Sigma^1_{1} \) and that \( U^{1}_{1} = (\forall \mathcal{F}(\exists y)A(\mathcal{F},y)) \). But \( U^{1}_{1}(\delta) \) is an extension of \( \check{U}^{1}_{1}(\delta) \), so \( \check{U}^{1}_{1}(\delta) \). Then Theorem 10 states that the desired function \( f \) exists. \( \Box \)

We shall prove Theorem 10 by proving a more general theorem:

**Theorem 12:** Suppose \( \check{U}^{1}_{1}(\delta) \gamma \sim \Gamma \wedge \Delta \) and each formula in \( \Gamma \wedge \Delta \) is a \( \Pi^1_{1} \) formula and each formula in \( \Pi \wedge \Delta \) is a \( \Sigma^1_{1} \) formula. Let \( \epsilon_1, \ldots, \epsilon_6 \) and \( \gamma_1, \ldots, \gamma_6 \) be the free variables in \( \Gamma \wedge \Delta \). Let \( X \) and \( Y \) be the \( \Sigma^1_{1} \) formulæ

\[
X = (\forall \mathcal{G})\gamma \exists \mathcal{A}(\neg C : \mathcal{C}(\mathcal{A}))
\]

and

\[
Y = (\forall \mathcal{G})\gamma \exists \mathcal{A}(\neg C : \mathcal{C}(\mathcal{A})).
\]

Then there is a PSPACE(\( \alpha, \gamma \)) predicate \( M \) so that

1. \( M \) is \( \Delta^1_{1}(\delta) \) defined by \( \check{U}^{1}_{1}(\delta) \) and
2. \( \check{U}^{1}_{1}(\delta) \) \( \sim \) Witness\( \exists \mathcal{F}(\alpha, \mathcal{C}, \gamma_7) \) \( \sim \) Witness\( \exists \mathcal{F}(\alpha, \mathcal{C}, \gamma_7) \) \( \sim \) \( M(\alpha, \mathcal{C}, \gamma_7) \).

**Proof:** of Theorem 10 from Theorem 12:

The hypothesis of Theorem 10 is that \( \check{U}^{1}_{1}(\delta) \gamma \sim \Gamma \wedge \Delta \). By the extension of Parikh's theorem to second order Bounded Arithmetic, there is a term \( t \) such that \( \check{U}^{1}_{1}(\delta) \gamma \sim (\forall \mathcal{F}(\exists y \leq t(F))A(\mathcal{F},y)) \). We now apply Theorem 12 with \( \Delta = (\exists y \leq t(F))A(\mathcal{F},y) \) and with \( \Gamma = \Pi \wedge \Delta = \emptyset \). Theorem 12 asserts that there is a PSPACE predicate \( M \) which is \( \Delta^1_{1}(\delta) \) defined by \( \check{U}^{1}_{1}(\delta) \) so that

\[
\check{U}^{1}_{1}(\delta) \gamma \sim \text{Witness}_{\Delta}(\exists \mathcal{F}(\alpha, \mathcal{C}, \gamma_7) \sim M(\alpha, \mathcal{C}, \gamma_7)).
\]

By the definition of Witness\( \exists \mathcal{F} \), this means that

\[
\check{U}^{1}_{1}(\delta) \gamma \sim \text{Witness}_{\Delta}(\exists \mathcal{F}(\alpha, \mathcal{C}, \gamma_7) \sim M(\alpha, \mathcal{C}, \gamma_7)).
\]
Now define

\[ f(\overline{z}) = (\mu y)\text{Witness}^{2d}_{\text{A}(f)}(x, M(\overline{z}, \overline{y})) \]

and

\[ B(\overline{z}, \overline{d}) \iff \text{Witness}^{2d}_{\text{A}(f)}(x, M(\overline{z}, \overline{d}), \lambda) \land (\forall y < d) \neg \text{Witness}^{2d}_{\text{A}(f)}(x, M(\overline{z}, \overline{y})). \]

Since \( \text{Witness}^{2d}_{\text{A}(f)}(x, \overline{z}, \overline{d}) \) is a PSPACE\( (\alpha) \) predicate and since \( \langle x \rangle M(\overline{z}) \) is a PSPACE predicate, \( f \) is readily seen to be polynomial space computable. Also, since \( \text{Witness}^{2d}_{\text{A}(f)} \) is a \( \Sigma^P_3 \)-formula and \( M \) is a \( \Delta^P_1 \)-defined predicate, \( B \) is a \( \Delta^P_1 \)-formula.

It now follows from Theorem 9.13(b) that conditions (1)-(3) 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 \( \text{FL}_2(\delta) \). Before we can prove Theorem 10, we need a lemma about the computational complexity of the inductive definition of Theorem 9.13.

Lemma 18: Let \( A(\overline{z}, \overline{z}) \) and \( B(\overline{a}, \overline{b}, \overline{z}, \overline{z}) \) be \( \Delta^P_1(\delta) \)-formulae of \( \overline{B}_2(\delta) \) where \( \alpha \) is a unary predicate variable. Let \( t(\overline{b}, \overline{z}) \) be a term. Let \( K(\overline{a}, \overline{b}, \overline{z}) \) be defined from \( A \) and \( B \) as in Theorem 9.13 by

\[
K(a, b, z) \iff \begin{cases} 
A(a, z) & \text{if } b = 0 \text{ and } a \leq t(b, z) \\
0 = 1 & \text{otherwise}
\end{cases}
\]

Then \( K(\overline{a}, \overline{b}, \overline{z}) \) is \( \Delta^P_1(\delta) \)-definable by \( \overline{B}_2(\delta) \). Furthermore, if \( A \) is in PSPACE\( (\gamma) \) and \( B \) is in PSPACE\( (\alpha, \gamma) \) then \( K \) is in PSPACE\( (\gamma) \) predicate.

Proof: The fact that \( K(\overline{a}, \overline{b}, \overline{z}) \) is \( \Delta^P_1(\delta) \)-defined by \( \overline{B}_2(\delta) \) is proved by the proof of Theorem 9.13. So we must prove \( K \) is in PSPACE\( (\gamma) \). To do this we specify an algorithm to compute \( K(\overline{a}, \overline{b}, \overline{z}) \).

Suppose \( b \neq 0 \) and \( a \leq t(b, z) \), then to compute \( K(z, b, z) \) we begin by computing \( B(\overline{a}, \overline{b}, \overline{z}, \overline{z}) \) with a PSPACE machine \( M_B \) with oracles for \( \alpha \) and \( \gamma \). However, we modify \( M_B \) so that whenever \( M_B \) would have queried the oracle of \( \alpha(\overline{z}) \), instead \( M_B \) saves its current state and begins to compute \( K(z, b, z) \). This process iterates until we wish to compute \( K(z, 0, z) \) for some \( z \). Then we just compute \( A(z, z) \) 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 \( \hat{\Delta}_{1}(\delta) \)-proof of \( \Gamma, \Pi \rightarrow A, \Delta \) such that \( P \) is free cut free and in free variable normal form. Hence, by Corollary 9.21, every formula in \( P \) is in \( \Sigma_{1}^{1}(\delta) \cap \Pi_{1}^{1}(\delta) \).

The proof \( P \) will generally contain a number of relational symbols \( \delta_{1}, \ldots, \delta_{i} \). These relational symbols are introduced with defining equations \( \delta_{i}(\xi, \delta) \rightarrow A_{i}(\delta, \delta) \) where \( A_{i} \in \Sigma_{1}^{1}(\delta) \). Thus the proof \( P \) requires auxiliary proofs \( P_{1}, \ldots, P_{i} \) of equivalences \( A_{i}(\delta, \delta) \equiv B_{i}(\delta, \delta) \) where each \( B_{i} \) is a \( \Pi_{1}^{1}(\delta) \)-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 \( \delta_{j} \) appearing in any of \( P_{i} \), \( j = 1, \ldots, k \), which is defined by \( \delta_{j}(\xi, \delta) \rightarrow A_{j}(\delta, \delta) \) there is a \( \Pi_{1}^{1}(\delta) \)-formula \( B_{j} \) and there are two proofs \( P_{j}, P_{k} \) of \( A_{j}(\delta, \delta) \rightarrow B_{j}(\delta, \delta) \) and \( B_{j}(\delta, \delta) \rightarrow A_{j}(\delta, \delta) \). In addition, we may assume that each proof \( P_{j}, P_{k} \), \( j = 1, \ldots, k \) is free cut free and that every formula appearing in \( P_{i} \), \( i = 1, \ldots, k \), is in \( \Sigma_{1}^{1}(\delta) \cap \Pi_{1}^{1}(\delta) \).

The proof of Theorem 12 is by induction on the total number of sequents in the proofs \( P_{1}, P_{2}, \ldots, P_{k} \). 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 \( P_{i} \) is the initial sequent

\[
\delta_{i}(\xi, \gamma) \rightarrow A_{i}(\xi, \gamma)
\]

where \( \Gamma = (\delta_{i}(\xi, \gamma)) \) and \( \Delta = (A_{i}(\xi, \gamma)) \). Then by assumption there is a proof \( P_{i} \) of

\[
\delta_{i}(\xi, \delta) \rightarrow A_{i}(\xi, \delta)
\]

where \( B_{i} \in \Pi_{1}^{1}(\delta) \). By the induction hypothesis, applied to \( P_{i} \), there is a \( \Sigma_{1}^{1}(\delta) \)-formula \( G \) which is \( \Delta_{1}^{1}(\delta) \)-defined by \( \hat{\Delta}_{1}(\delta) \) such that

\[
\hat{\Delta}_{1}(\delta) \equiv \text{Witness}_{\Sigma_{1}^{1}}(1, (1) G(x, x, \delta), (x, \delta)) \lor \\
\text{Witness}_{\Sigma_{1}^{1}}(2, (1) G(x, x, \delta), (x, \delta))
\]

Since

\[
\hat{\Delta}_{1}(\delta) \equiv \text{Witness}_{\Sigma_{1}^{1}}(a, x, \delta) \lor B_{j}(x, \delta)
\]

and \( \hat{\Delta}_{1}(\delta) \equiv \neg B_{j} \lor \neg A_{j} \), we have
Deﬁnable Functions of Second Order Bounded Arithmetic

\[ \varnothing (\alpha, \overline{\beta}, \overline{\gamma}) \Rightarrow \text{Witness} \varnothing (\alpha, \overline{\beta}, \overline{\gamma}) \Rightarrow G(x, \overline{\beta}, \overline{\gamma}). \]

So set \( M \) to be the PSPACE predicate deﬁned by \( M(x, \overline{\beta}, \overline{\gamma}) \iff G(1, x, \overline{\beta}, \overline{\gamma}) \). Now since \( \varnothing (\alpha, \overline{\beta}, \overline{\gamma}) \) is atomic, we have

\[ \varnothing (\alpha, \overline{\beta}, \overline{\gamma}) \Rightarrow \text{Witness} \varnothing (\alpha, \overline{\beta}, \overline{\gamma}) \Rightarrow \text{Witness} \varnothing (\alpha, \overline{\beta}, \overline{\gamma}) \Rightarrow M(x, \overline{\beta}, \overline{\gamma}). \]

This proves the theorem for the case where \( P \) is a single initial sequent of the form \( \varnothing (\alpha, \overline{\beta}, \overline{\gamma}) \Rightarrow A (\alpha, \overline{\beta}, \overline{\gamma}) \). 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 \( \varnothing (\alpha, \overline{\beta}, \overline{\gamma}) \) appearing in \( P \) is a PSPACE predicate.

Next we consider the case where \( P \) does contain one or more inferences. We shall henceforth make the simplifying assumption that \( \Pi \) and \( \Delta \) are the empty cedent. As in the proof of Theorem 5.5 this involves no loss of generality since \((\neg \text{left})\) and \((\neg \text{right})\) inferences can be used to move formulae from side to side and since each inference has a dual. The argument splits into 18 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)-(9). Omitted.

Case (10): \( (\lor \text{left}) \). Suppose the last inference of \( P \) is

\[ B, \Gamma^* \rightarrow \Delta \]

\[ C, \Gamma^* \rightarrow \Delta \]

\[ BV C, \Gamma^* \rightarrow \Delta \]

Let \( D \) be the formula \( B \land (\neg \Delta^*) \), let \( E \) be \( C \land (\neg \Delta^*) \) and let \( F \) be \( (BV C) \lor (\neg \Delta^*) \).

The induction hypothesis is that there are FSPACE predicates \( G \) and \( H \) which are \( \Delta^*(\delta) \)-deﬁned by \( \varnothing (\delta) \) such that

\[ \varnothing (\delta) \Rightarrow \text{Witness} \varnothing (\alpha, \overline{\beta}, \overline{\gamma}) \Rightarrow \text{Witness} \varnothing (\alpha, \overline{\beta}, \overline{\gamma}) \Rightarrow G(x, \overline{\beta}, \overline{\gamma}). \]

\[ \varnothing (\delta) \Rightarrow \text{Witness} \varnothing (\alpha, \overline{\beta}, \overline{\gamma}) \Rightarrow \text{Witness} \varnothing (\alpha, \overline{\beta}, \overline{\gamma}) \Rightarrow H(x, \overline{\beta}, \overline{\gamma}). \]

Define \( M \) by
\[ M(x, z, \alpha, \gamma) \iff \begin{cases} G(x, z, < \beta(1, \alpha)>, \beta(2, \alpha) > \gamma) \\
\text{if Witness}_{\beta}^2(\alpha, 1, \beta) \end{cases} \]

Clearly \( M \) is a PSPACE(\( \alpha, \gamma \)) predicate and is \( \Delta^1_k(\delta) \)-definable by \( \bar{B}_2(\delta) \) since \( G, H \) and Witness_{\beta}^2 are. It is now easy to see that

\[ \bar{B}_2(\delta) : \text{Witness}_{\beta}^2(\alpha, \beta, \gamma) \supset \text{Witness}_{\beta}^2(\{x\} M(x, z, \alpha, \gamma), \beta, \gamma) \]

Cases (4)-(18) Omitted.

Case (14k) (second order \( \exists \)-left). Suppose the last inference of \( P \) is

\[ \frac{B(\delta)_1, \ldots \rightarrow \Delta}{(\exists \phi) B(\phi)_1, \ldots \rightarrow \Delta} \]

where \( \beta \) and \( \phi \) are \( k \)-ary predicate variables and \( \beta \) is the eigenvariable and must not appear in the lower sequent.

Let \( D \) be the formula \( B(\beta)_1(A\gamma^*) \) and let \( E \) be \( (\exists \phi) B(\phi)_1(A\gamma^*) \). The induction hypothesis is that there is a PSPACE(\( \alpha, \beta, \gamma \)) predicate \( G \) which is \( \Delta^1_k(\delta) \)-defined by \( \bar{B}_2(\delta) \) such that

\[ \bar{B}_2(\delta) : \text{Witness}_{\beta}^2(\alpha, \beta, \gamma) \supset \text{Witness}_{\beta}^2(\{x\} G(x, z, \alpha, \beta, \gamma), \beta, \gamma) \]

Note we can omit \( \beta \) from the superscript on the lefthand side of this implication since \( \beta \) does not appear in \( \Delta \).

Let \( M \) be the predicate \( \Delta^1_k(\delta) \)-defined by

\[ M(x, z, \alpha, \gamma) \iff G(x, z, \beta(2, \alpha), \text{ARY}(\beta(1, \alpha)), \gamma) \]

Clearly, \( M \) is in PSPACE(\( \alpha, \gamma \)) since \( G \) is in PSPACE(\( \alpha, \beta, \gamma \)). Furthermore it is easy to see that

\[ \bar{B}_2(\delta) : \text{Witness}_{\beta}^2(\alpha, \beta, \gamma) \supset \text{Witness}_{\beta}^2(\{x\} M(x, z, \alpha, \gamma), \beta, \gamma) \]


Case (15): (second order $\exists$-right). Suppose the last inference of $P$ is

$$
\Gamma \Rightarrow B(\psi), \Delta^*
$$
$$
\Gamma \Rightarrow (\exists \phi) B(\phi), \Delta^*
$$

where $\phi$ is a $k$-ary predicate variable and $V$ is the abstract $\{y_1, \ldots, y_k\} A(y_1, \ldots, y_k, \gamma)$
where $A$ is a $\Sigma^B_k(\delta)$-formula.

Let $D$ be the formula $B(V)v(\nu \Delta^*)$ and let $E$ be $(\exists \phi) B(\phi)v(\nu \Delta^*)$. The induction hypothesis is that there is a $\text{PSPACE}(\alpha, \gamma)$ predicate $G$ which is $\Delta^L_k(\delta)$-defined by $\hat{\nu}_{\delta}(\delta)$ such that

$$
\hat{\nu}_{\delta}(\delta)\text{-Witness}^\epsilon_{\hat{\nu}_{\delta}}(\alpha, \gamma, \hat{\nu}_{\delta}(\delta))\supseteq \text{Witness}^\epsilon_{\hat{\nu}_{\delta}}(\beta) G(z, \alpha, \gamma, \hat{\nu}_{\delta}(\delta)).
$$

Let $M$ be the predicate $\Delta^L_k(\delta)$-defined in $\hat{\nu}_{\delta}(\delta)$ by

$$
M(z, \alpha, \gamma) \iff \begin{cases} 
G(z, \alpha, \gamma) & \text{if } z = \langle 2, z \rangle \\
A(z, \alpha, \gamma) & \text{if } z = \langle 1, y_1, \ldots, y_k \rangle \\
0 = 1 & \text{otherwise}
\end{cases}
$$

In other words, $\{z\} M$ is equal to $\langle \text{DELETE}_{\delta}((\beta) A), \langle z \rangle G \rangle$. It now follows from Lemma 9 that

$$
\hat{\nu}_{\delta}(\delta)\text{-Witness}^\epsilon_{\hat{\nu}_{\delta}}(\alpha, \gamma, \hat{\nu}_{\delta}(\delta))\supseteq \text{Witness}^\epsilon_{\hat{\nu}_{\delta}}(\beta) M(z, \alpha, \gamma, \hat{\nu}_{\delta}(\delta)).
$$

It remains to show that $M$ is a $\text{PSPACE}(\alpha, \gamma)$ predicate. Since $G$ is a $\text{PSPACE}(\alpha, \gamma)$ predicate by the induction hypothesis, it suffices to show that $A$ is a $\text{PSPACE}(\gamma)$ predicate. But this follows from the fact that $A$ is in $\Sigma^B_k(\delta)$ and, as we remarked earlier, every relational appearing in $A$ is a $\text{PSPACE}(\gamma)$ predicate.

Case (16): ($\Sigma^L_k(\delta)$-PIND). Suppose the last inference of $P$ is

$$
B(\langle 1, a \rangle), \Gamma^* \Rightarrow B(a), \Delta^*
$$
$$
B(0), \Gamma^* \Rightarrow B(t), \Delta^*
$$

where $B$ is a $\Sigma^L_k(\delta)$-formula and $a$ is the eigenvariable and does not appear in the lower sequent. We shall assume that $B(0)$ is in $\Gamma$ and $B(t)$ is in $\Delta$. The other cases are easier and are omitted.
Let $D$ be the formula $B(1,x)\land (\forall \Delta^*)$, and let $E(\bar{x},a)$ be $B(\bar{x})\land (\forall \Delta^*)$. Let $F$ be $B(0)\land (\forall \Delta^*)$ and let $A$ be $B(\bar{x})\land (\forall \Delta^*)$. The induction hypothesis is that there is a PSPACE($\bar{\gamma}$) predicate $G$ such that $G$ is $\Delta^{1,4}(\bar{\delta})$-defined by $\tilde{B}_1(\delta)$ and such that

$$\tilde{B}_1(\delta)\vdash \text{Witness}\tilde{B}_2(\bar{x},a,\bar{\gamma}) \supset \text{Witness}\tilde{B}_2(\bar{x},a,\bar{\gamma}) \supset G(x,\bar{x},a,\bar{\gamma},\bar{x},a,\bar{\gamma}).$$

By Lemma 9, there is a term $s(\bar{x},a)$ such that

$$\tilde{B}_2(\delta)\vdash (\exists 0 \leq s(\bar{x},a))(o(x) \rightarrow \beta(x)) \supset \text{Witness}\tilde{B}_2(\bar{x},a,\bar{\gamma}) \supset \text{Witness}\tilde{B}_2(\bar{x},a,\bar{\gamma}).$$

By Lemma 13, there is a $\Delta^{1,4}(\bar{\delta})$-definable predicate $K$ of $\tilde{B}_2(\delta)$ which satisfies

$$K(x,\bar{x},a,\bar{\gamma}) \iff \begin{cases} 0=1 & \text{if } x > s(\bar{x},a) \\ o(x) & \text{if } b=0:0 \leq s(\bar{x},a) \\ K(x,\bar{x},[s],\bar{\gamma}) & \text{if } x \leq s(\bar{x},a) \\ G(x,\bar{x},a,\beta(2,[s],a,\bar{\gamma}),\beta(2,a) > \bar{\gamma}) & \text{otherwise} \end{cases}$$

Furthermore, by Lemma 13, $K$ is in PSPACE($\bar{\gamma}$). From the definition of $K$ it is readily seen that

$$\tilde{B}_2(\delta)\vdash \text{Witness}\tilde{B}_2(\bar{x},a,\bar{\gamma}) \supset \text{Witness}\tilde{B}_2(\bar{x},a,\bar{\gamma}) \supset \text{Witness}\tilde{B}_2(\bar{x},a,\bar{\gamma}) \supset G(x,\bar{x},a,\bar{\gamma}).$$

Hence it follows by $\Sigma^{1,4}(\bar{\delta})$-$\text{PIND}$ that

$$\tilde{B}_2(\delta)\vdash \text{Witness}\tilde{B}_2(\bar{x},a,\bar{\gamma}) \supset \text{Witness}\tilde{B}_2(\bar{x},a,\bar{\gamma}) \supset G(x,\bar{x},a,\bar{\gamma}).$$

So we define $M(x,\bar{x},a,\bar{\gamma})$ by
Definable Functions of Second Order Bounded Arithmetic

\[ M(x,\bar{x},\alpha,\bar{a}) \iff K(x,\bar{x},\alpha,\bar{a}) \]

and \( M \) satisfies the conditions of Theorem 12.

Q.E.D. \( \Box \)

10.6. Only EXPTIME is \( \Sigma^b_1 \)-definable in \( V_2 \).

Theorem 1 asserted that every EXPTIME function of polynomial growth rate is \( \Sigma^b_1 \)-definable by \( V_2 \). The converse is also true. Since the proof of the converse to Theorem 1 is very similar to the arguments in \( \S 10.5 \) concerning \( \Sigma^b_1 \)-definable functions of \( V_2 \) and \( \bar{V}_2(d) \) we shall merely state the results without giving detailed proofs.

**Theorem 14:** Suppose \( A(\bar{x},d) \) is a \( \Sigma^b_1(\delta) \)-formula where \( \bar{x} \) and \( d \) are all the free variables of \( A \). Also suppose \( \bar{V}_2(d) \vdash (\forall \bar{x})(\exists y)A(\bar{x},y) \). Then there is a \( \Delta^b_1(\delta) \)-formula \( B \), a term \( t \) and a function \( f \) so that

1. \( \bar{V}_2(d) \vdash (\forall \bar{x})(\forall y)(B(\bar{x},y) \supset A(\bar{x},y)) \)
2. \( \bar{V}_2(d) \vdash (\forall \bar{x})(\exists y \leq t)B(\bar{x},y) \)
3. \( \bar{V}_2(d) \vdash (\forall \bar{x})(\forall y)(B(\bar{x},y) \supset B(t,\bar{x},y) \supset \exists x \supset y = x) \)
4. For all \( \bar{x}, N := B(\bar{x},f(\bar{x})) \)
5. \( f \) is an EXPTIME function

Hence, \( f \) is an EXPTIME function which is \( \Sigma^b_1(\delta) \)-definable in \( \bar{V}_2(d) \) and for all \( \bar{x}, A(\bar{x},f(\bar{x})) \) is true.

The converse to Theorem 1 is an immediate corollary of Theorem 14:

**Corollary 15:** Suppose \( A(\bar{x},d) \) is a \( \Sigma^b_1 \)-formula where \( \bar{x} \) and \( d \) are all the free variables of \( A \). Also suppose \( V_2 \vdash (\forall \bar{x})(\exists y)A(\bar{x},y) \). Then there is an EXPTIME function \( f \) such that for all \( \bar{x}, N := A(\bar{x},f(\bar{x})) \).

As before, the proof of Theorem 14 is based on a more complicated theorem:

**Theorem 16:** Suppose \( \bar{V}_2(d) \vdash \Gamma, \Pi \rightarrow \Lambda, \Delta \) and each formula in \( \Gamma \cup \Delta \) is a \( \Sigma^b_1(\delta) \)-formula and each formula in \( \Pi \cup \Lambda \) is a \( \Pi^b_1(\delta) \)-formula. Let \( c_1, \ldots, c_p \) and \( \gamma_1, \ldots, \gamma_q \) be the free variables in \( \Gamma, \Pi \rightarrow \Lambda, \Delta \). Let \( X \) and \( Y \) be the \( \Sigma^b_1 \)-formulae
$X = (\forall \Delta)^c A \land (\neg C : C \subseteq A)$

and

$Y = (\forall \Delta)^c \forall C : C \subseteq \Pi$.

Then there is an EXPTIME($\alpha, \gamma$) predicate $M$ so that

1. $M$ is $\Delta^A_1(\delta)$-defined by $\hat{P}_2(\delta)$ and
2. $\hat{P}_2(\delta)$—Witness $\gamma^x(\alpha, \mathfrak{C}, \gamma) \vdash \text{Witnesses } \hat{P}_2(\delta)$.

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 $\Sigma^A_1(\delta)$-IND inference. In this case, instead of using Lemma 13 we use Lemma 17:

**Lemma 17:** Let $A(x, \mathfrak{C}, \gamma)$ and $B(a, b, \mathfrak{C}, \alpha, \gamma)$ be $\Delta^A_1(\delta)$-formulae of $\hat{P}_2(\delta)$ where $\alpha$ is a unary predicate variable. Let $l(b, \mathfrak{C})$ be a term with only the free variables $b$ and $\mathfrak{C}$ as indicated. Let $K(a, b, \mathfrak{C}, \gamma)$ be defined from $A$ and $B$ as in Theorem 9.14 by:

$$K(a, b, \mathfrak{C}, \gamma) \iff \begin{cases} A(a, \mathfrak{C}, \gamma) & \text{if } b=0 \text{ and } a \leq l(b, \mathfrak{C}) \\ 0=1 & \text{if } a > l(b, \mathfrak{C}) \\ B(a, b, \mathfrak{C}, x)K(x, b+1, \mathfrak{C}, \gamma) & \text{otherwise} \end{cases}$$

Then $K(a, b, \mathfrak{C}, \gamma)$ is $\Delta^A_1$-definable by $\hat{P}_2(\delta)$. Furthermore, if $A$ is in EXPTIME($\alpha, \gamma$) and $B$ is in EXPTIME($\gamma, \gamma$) then $K$ is in EXPTIME($\gamma, \gamma$).

**Proof:** The proof of Theorem 9.14 shows that $K(a, b, \mathfrak{C}, \gamma)$ in $\Delta^A_1(\delta)$-defined by $\hat{P}_2(\delta)$. If $A$ is EXPTIME($\gamma$) computable and $B$ is EXPTIME($\alpha, \gamma$) computable, then the straightforward algorithm for computing $K(a, b, \mathfrak{C}, \gamma)$ is an EXPTIME($\gamma, \gamma$)-algorithm.

Q.E.D. □

10.7. A Corollary about $\text{NEXPTIME}$ co-$\text{NEXPTIME}$.

**Definition:** $\text{NEXPTIME}$ is the set of predicates which are recognized by a non-deterministic exponential time Turing machine. The set co-$\text{NEXPTIME}$ is the set of predicates whose complements are in $\text{NEXPTIME}$. 
Proposition 18: A predicate $Q(y)$ is in NEXPTIME if there is a formula $A \in \Sigma^1_{1,k}$ such that

$$Q(y) \iff N = A(f).$$

Proof: By Corollary 9.17, every $\Sigma^1_{1,k}$-formula $A(f)$ is equivalent to a formula of the form $(\exists b)B(x,\phi)$ where $B \in \Sigma^1_{1,k}$. By Lemma 9, there is a term $s_d(x)$ so that the value of $B(f, \phi)$ only depends on the values of $\phi(y)$ for $y \leq s_d(f)$. Thus a $\Sigma^1_{1,k}$-formula $A(f)$ can be evaluated in non-deterministic exponential time by first guessing the values of $\phi(y)$ for all $y \leq s_d(f)$ and then evaluating $B(f, \phi)$.

Conversely, it follows from the methods of §10.1 that every NEXPTIME predicate $Q(y)$ can be expressed by a $\Sigma^1_{1,k}$-formula $A(f)$. If $M$ is a non-deterministic Turing machine which computes $Q(f)$ in time $2^{O(n)}$, let $A(f)$ be $(\exists \lambda)B(\text{w}, \lambda, n, n^2)\iff f$.

Q.E.D. □

Corollary 19:

(a) If $A(f)$ is any formula and $U^l_2$ proves $A(f)$ is equivalent to a $\Sigma^1_{1,k}$ and a $\Pi^1_{1,k}$-formula then $A(f)$ represents a predicate in PSPACE. In other words, if $U^l_2$ proves $A$ is in NEXPTIME/F∞-NEXPTIME then $A \in$ PSPACE.

(b) If $A(f)$ is any formula and $V^l_2$ proves $A(f)$ is equivalent to a $\Sigma^1_{1,k}$ and a $\Pi^1_{1,k}$-formula then $A(f)$ represents a predicate in EXPTIME. In other words, if $V^l_2$ proves $A$ is in NEXPTIME/F∞-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 5.19. □

Corollary 19 also holds for the theories $\overline{\mathcal{L}}^l_2$ and $\mathcal{L}^l_2$.

10.8. Variations, Complications and Open Questions.

Some questions concerning second order Bounded Arithmetic which have not been resolved include:

1. Is $V^l_2$ equivalent to $U^l_2$?
2. Is $U^l_2$ equivalent to $\overline{U}^l_2$?
3. Is $U^l_2$ a conservative extension of $S_2$?
4. Is $U^l_2$ a conservative extension of $S_2$?

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 $U^1_2 = V^1_2$ then PSPACE = EXPTIME. Also, if $U^1_2(\delta) = V^1_2(\delta)$ then PSPACE = EXPTIME.

Proof: By Theorems 1, 4, 10 and 14. □

However, there seems to be no reason why $U^1_2$ could not be a conservative extension of $S^1_2$. There is no evidence that this would imply $P = \text{PSPACE}$, for instance.

A topic for further research would be to investigate the theories $U^j_2$ and $V^j_2$ for $j > 1$. It would be nice to establish what functions can be $\Sigma^i_2$-defined in these theories. It appears that the $\Sigma^i_2$-defined functions of $V^2_2$ are precisely the functions at the $i$-th level of the exponential time hierarchy. That is, $V^2_2$ can $\Sigma^i_2$-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 $U^2_2$ 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 $U^2_2$ can $\Sigma^i_2$-define 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 $\Sigma^i_2$-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 $\phi$ has bounded domain iff there is a $z$ such that when $x_z > z$ for some $x_z$ then $\phi(z)$ does not hold. Likewise, a function $\lambda$ has bounded domain iff there is a $z$ such that when some $x_z > z$, $\lambda(z) = 0$.

We change the second order language so that the second order predicate variables are $\alpha^i$ and $\phi^i$ and the second order function variables are $\lambda^i$ and $\lambda^i$ where $z$ and $i$ are arbitrary terms. Let $\mathcal{F}$ be a list of new variables not appearing in $z$. Then second order Bounded Arithmetic contains the new axioms

$$(\forall \mathcal{F})(\forall \alpha^i)(\forall x_z > z)(\neg \alpha^i(z))$$

$$(\forall \mathcal{F})(\forall \lambda^i)(\forall x_z > z)(\lambda^i(x) = 0)$$

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 \( \Sigma^1_0\)-BCA axioms) are
\[
(\forall \bar{y})(\forall \bar{x})(\forall r)(\exists \bar{y})(\forall \bar{y} \leq r)\lambda(\bar{y}) \rightarrow A(\bar{y}, \bar{x}, \bar{y}^{r})
\]
where \( A \in \Sigma^1_0 \). The \( \Sigma^1_0\)-BFCA, the bounded domain function comprehension axioms are defined similarly. We leave it to the reader to formulate the bounded domain comprehension inferences.

Let \( U^1_1(BD) \) and \( V^1_1(BD) \) be the theories which use bounded domain predicate and function variables, have the \( \Sigma^1_0\)-FIND and \( \Sigma^1_0\)-IND (respectively) axioms, and have the \( \Sigma^1_0\)-comprehension axioms. So \( U^1_1(BD) \) and \( V^1_1(BD) \) are similar to \( U^1_1 \) and \( V^1_1 \) except they are restricted to using only bounded domain second order variables. It turns out that the same functions are \( \Sigma^1_0\)-definable in \( U^1_1(BD) \) and \( V^1_1(BD) \) as in \( U^1_1 \) and \( V^1_1 \) 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 \( \bar{U}_1(BD) \) and \( \bar{V}_1(BD) \) are defined to be \( U^1_1(BD) \) and \( V^1_1(BD) \), 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 \( \bar{U}_1(BD) \) and \( \bar{V}_1(BD) \) are conservative extensions of \( U^1_1(BD) \) and \( V^1_1(BD) \) 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 \( S^2_2 \) and \( S^2_2 \) are predicative.

Second order bounded domain Bounded Arithmetic is also predicative. To show this it suffices to interpret \( \bar{U}_1(BD) \) in the first order theory \( S^2_2 \). So let \( M \) be a model of \( S^2_2 \), we construct from \( M \) a model \( N \) for \( \bar{U}_1(BD) \). \( N \) 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 \( \pi \in N_1 \) then we interpret \( \alpha(\pi) \) in \( S^2_2 \) as
\[
\beta(\langle \pi, \alpha \rangle) \forall \theta
\]
where \( \langle \pi, \alpha \rangle \) is the sequence coding \( z_1, \ldots, z_n \) and satisfies
\[
UniqSeq(\langle \pi, \alpha \rangle) \land 0(\langle \pi, \alpha \rangle, n) \land (\exists i < n)(\beta(i+1, \langle \pi, \alpha \rangle) = z_i).
\]
By the results of Chapter 2, it is clear that for each \( n \geq 0 \) the map \( \beta \rightarrow \langle \pi, \alpha \rangle \) is \( \Sigma^1_0 \)-defined by \( S^2_2 \). Hence the interpretation of \( \alpha(\pi) \) is well-defined.

Next define \( I(\theta) \) to specify an initial segment of \( M \) satisfying
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 $2^m$ exists in $M$. Since, $2^{2^m} = (2^m)^2$, $I$ 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 $N = \langle N_1, N_2 \rangle$ is a model of $\overline{U}_2^{\Sigma_1^0}(BD)$. This is because the $\Pi^1_1$-$BCA$ comprehension axioms can be proved using the $\Pi^1_1$-$IND$ axioms of $S_2$. Since this is straightforward, we omit the proof.

The above shows that $\overline{U}_2^{\Sigma_1^0}(BD)$ can be interpreted in $S_2$. It remains to show that $\overline{U}_2(BD)$ is interpretable in $\overline{U}_2^{\Sigma_1^0}(BD)$. The fact that $\overline{U}_2(BD)$ can be locally interpreted in $\overline{U}_2^{\Sigma_1^0}(BD)$ 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 $\overline{U}_2(BD)$ can be globally interpreted in $\overline{U}_2^{\Sigma_1^0}(BD)$ follows from a technique due to Wilkie, see Pudlák [22].

If $M$ is not closed under exponentiation, the above construction will actually yield a model $N$ of $\overline{U}_2$. Hook [16] uses the assumption $(\exists y)(\forall z)(x < y)$ as a predicative assumption. Hence, if we accept Hook’s axiom as predicative, the (unbounded domain) second order theories $U_2$ of Bounded Arithmetic are predicative.

As a corollary to the above discussion we deduce that the PSPACE and EXPTIME functions can be predicatively defined.
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. Renayre in a handwritten manuscript titled "A conservation result for systems of Bounded Arithmetic" has examined a strong form of the $\Sigma^b_1$-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.


Martin Dowd, personal communication.


215


<table>
<thead>
<tr>
<th>Symbol</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>$</td>
<td>a</td>
</tr>
<tr>
<td>$b$</td>
<td>2, 28</td>
</tr>
<tr>
<td>$\lfloor x \rfloor$</td>
<td>2</td>
</tr>
<tr>
<td>$\lfloor x \rfloor$</td>
<td>2</td>
</tr>
<tr>
<td>$N$</td>
<td>6</td>
</tr>
<tr>
<td>$B'$</td>
<td>7</td>
</tr>
<tr>
<td>$B''$</td>
<td>8</td>
</tr>
<tr>
<td>$[E]$</td>
<td>8</td>
</tr>
<tr>
<td>$P(\text{T}(C))$</td>
<td>9, 23</td>
</tr>
<tr>
<td>$P$</td>
<td>10</td>
</tr>
<tr>
<td>$PRED(C')$</td>
<td>13, 22</td>
</tr>
<tr>
<td>$PBV(C')$</td>
<td>13, 24</td>
</tr>
<tr>
<td>$PBV(C')$</td>
<td>13, 24</td>
</tr>
<tr>
<td>$LBV(C')$</td>
<td>14</td>
</tr>
<tr>
<td>$LBV(C)$</td>
<td>14</td>
</tr>
<tr>
<td>$\Delta k'$</td>
<td>14</td>
</tr>
<tr>
<td>$\Sigma k'$</td>
<td>14</td>
</tr>
<tr>
<td>$\Pi k'$</td>
<td>14</td>
</tr>
<tr>
<td>$\Omega k'$</td>
<td>14</td>
</tr>
<tr>
<td>$PH$</td>
<td>14</td>
</tr>
<tr>
<td>$\Sigma k$</td>
<td>20, 29</td>
</tr>
<tr>
<td>$\Pi k$</td>
<td>20, 29</td>
</tr>
<tr>
<td>$\Delta k$</td>
<td>20, 29</td>
</tr>
<tr>
<td>$\Omega k(\emptyset)$</td>
<td>22</td>
</tr>
<tr>
<td>$\Delta k(\emptyset)$</td>
<td>22</td>
</tr>
<tr>
<td>$\Sigma k(\emptyset)$</td>
<td>22</td>
</tr>
<tr>
<td>$\Pi k(\emptyset)$</td>
<td>22</td>
</tr>
<tr>
<td>$PH(\emptyset)$</td>
<td>22</td>
</tr>
<tr>
<td>$\omega f'$</td>
<td>22</td>
</tr>
<tr>
<td>$\varphi(\theta)$</td>
<td>25</td>
</tr>
<tr>
<td>$\psi(\theta)$</td>
<td>25</td>
</tr>
<tr>
<td>$\xi(\theta)$</td>
<td>25</td>
</tr>
<tr>
<td>$\psi(\theta)$</td>
<td>25</td>
</tr>
<tr>
<td>$\varphi(\theta)$</td>
<td>25</td>
</tr>
<tr>
<td>$\rightarrow$</td>
<td>28</td>
</tr>
<tr>
<td>$#k$</td>
<td>28</td>
</tr>
<tr>
<td>$S(\theta)b$</td>
<td>29</td>
</tr>
<tr>
<td>$I\theta$</td>
<td>29</td>
</tr>
</tbody>
</table>

**Symbols**

- Basic: 30
- $\Psi_{-\text{IND}}$: 31, 72
- $\Psi_{-\text{PIN}}$: 31, 72
- $\Psi_{-\text{IND}}$: 32, 72
- $S_k$: 32, 72
- $S_k$: 32
- $T_k$: 32, 72
- $T_k$: 32
- $S_{k-1}^{-1}$: 32
- $\Sigma_{k-\text{define}}$: 33
- $\Sigma_k(f)$: 33
- $\Pi_k(f)$: 33
- $\Delta_k(f)$: 33
- $\Delta_k$: 35
- $\text{Power}^2$: 39
- $\text{Exp}$: 39
- $\text{Mod}^2$: 40
- $LSP$: 40
- $\text{MSP}$: 40
- $\text{Bit}$: 41
- $(\# x \leq t)(\ldots)$: 46
- $(\mu y \leq t)(\ldots)$: 47
- $\text{SqBd}$: 50
- $\rightarrow$: 51
- $\Sigma_k(\text{AS})$: 53
- $\Pi_k(\text{AS})$: 53
- $\Psi_{-\text{MIN}}$: 56
- $\Psi_{-\text{LMIN}}$: 56
- $\text{LK}$: 71
- $\text{LKB}$: 71
- $\sigma$: 77
- Witness: 86
- $\text{V}$: 89
- $\text{A}$: 89
- $\Leftarrow\ldots\Rightarrow$: 69
- $\eta$: 100
- $\text{PHP}(f)$: 102
- $PV$: 104
<table>
<thead>
<tr>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>NEXPTIME</td>
<td>207</td>
</tr>
<tr>
<td>$\Sigma^p_{1,k} - BCA$</td>
<td>210</td>
</tr>
<tr>
<td>$\Sigma^m_{1,k} - BFCA$</td>
<td>210</td>
</tr>
<tr>
<td>$U_1^\lambda(BD)$</td>
<td>210</td>
</tr>
<tr>
<td>$V_1^\lambda(BD)$</td>
<td>210</td>
</tr>
<tr>
<td>$\tilde{U}_1^\lambda(BD)$</td>
<td>210</td>
</tr>
<tr>
<td>$\tilde{V}_1^\lambda(BD)$</td>
<td>210</td>
</tr>
<tr>
<td>Term</td>
<td>Page</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>abstract</td>
<td>182</td>
</tr>
<tr>
<td>alternative sentence</td>
<td>53</td>
</tr>
<tr>
<td>ancestor</td>
<td>74</td>
</tr>
<tr>
<td>antecedent</td>
<td>67</td>
</tr>
<tr>
<td>arithmetic formula</td>
<td>19</td>
</tr>
<tr>
<td>arity</td>
<td>22, 180</td>
</tr>
<tr>
<td>atomic abstract</td>
<td>163</td>
</tr>
<tr>
<td>atomic formula</td>
<td>65, 127</td>
</tr>
<tr>
<td>auxiliary formula</td>
<td>70</td>
</tr>
<tr>
<td>bound variable</td>
<td>67, 160</td>
</tr>
<tr>
<td>bounded counting</td>
<td>46, 190</td>
</tr>
<tr>
<td>bounded domain</td>
<td>209</td>
</tr>
<tr>
<td>bounded formula</td>
<td>20, 29, 71, 160</td>
</tr>
<tr>
<td>bounded proof</td>
<td>71</td>
</tr>
<tr>
<td>bounded quantifier</td>
<td>20, 29, 166</td>
</tr>
<tr>
<td>bounded sequent</td>
<td>71</td>
</tr>
<tr>
<td>bounded theory</td>
<td>152</td>
</tr>
<tr>
<td>bounding axiom</td>
<td>190</td>
</tr>
<tr>
<td>cedent</td>
<td>67, 131</td>
</tr>
<tr>
<td>closed formula</td>
<td>66</td>
</tr>
<tr>
<td>closed term</td>
<td>66</td>
</tr>
<tr>
<td>closed under substitution</td>
<td>163</td>
</tr>
<tr>
<td>collapses</td>
<td>15</td>
</tr>
<tr>
<td>comprehension axiom</td>
<td>163, 164, 210</td>
</tr>
<tr>
<td>comprehension rule</td>
<td>163, 164</td>
</tr>
<tr>
<td>counting, length-bounded</td>
<td>46</td>
</tr>
<tr>
<td>cut formula</td>
<td>70</td>
</tr>
<tr>
<td>cut free proof</td>
<td>75</td>
</tr>
<tr>
<td>descendant</td>
<td>74</td>
</tr>
<tr>
<td>direct ancestor</td>
<td>74</td>
</tr>
<tr>
<td>direct descendant</td>
<td>74</td>
</tr>
<tr>
<td>distance</td>
<td>171</td>
</tr>
<tr>
<td>sign variable</td>
<td>70, 72, 161</td>
</tr>
<tr>
<td>elimination inference</td>
<td>75</td>
</tr>
<tr>
<td>endsequent</td>
<td>70</td>
</tr>
<tr>
<td>equality axiom</td>
<td>70</td>
</tr>
<tr>
<td>essentially proves</td>
<td>110</td>
</tr>
<tr>
<td>exponentiation</td>
<td>32, 39</td>
</tr>
<tr>
<td>father</td>
<td>117</td>
</tr>
<tr>
<td>feasible function</td>
<td>2</td>
</tr>
<tr>
<td>first order formula</td>
<td>160</td>
</tr>
<tr>
<td>formula</td>
<td>66, 128</td>
</tr>
<tr>
<td>free cut</td>
<td>74</td>
</tr>
<tr>
<td>free cut free proof</td>
<td>75</td>
</tr>
<tr>
<td>free formula</td>
<td>74</td>
</tr>
<tr>
<td>free mix</td>
<td>171</td>
</tr>
<tr>
<td>free variable</td>
<td>67, 160</td>
</tr>
<tr>
<td>free variable normal form</td>
<td>76</td>
</tr>
<tr>
<td>function oracle</td>
<td>22</td>
</tr>
<tr>
<td>function space</td>
<td>22, 101</td>
</tr>
<tr>
<td>functional</td>
<td>22</td>
</tr>
<tr>
<td>grade</td>
<td>171</td>
</tr>
<tr>
<td>induction axiom</td>
<td>31, 178</td>
</tr>
<tr>
<td>induction inference</td>
<td>72</td>
</tr>
<tr>
<td>inessential cut</td>
<td>75</td>
</tr>
<tr>
<td>inference</td>
<td>67, 70</td>
</tr>
<tr>
<td>initial sequent</td>
<td>70, 72</td>
</tr>
<tr>
<td>jump</td>
<td>157</td>
</tr>
<tr>
<td>left rank</td>
<td>171</td>
</tr>
<tr>
<td>length-bounded counting</td>
<td>46</td>
</tr>
<tr>
<td>level</td>
<td>171</td>
</tr>
<tr>
<td>limited iteration</td>
<td>8, 23</td>
</tr>
<tr>
<td>limited iteration on notation</td>
<td>104</td>
</tr>
<tr>
<td>limited recursion</td>
<td>191</td>
</tr>
<tr>
<td>logarithmically bounded quantification</td>
<td>13</td>
</tr>
<tr>
<td>logical axiom</td>
<td>70</td>
</tr>
<tr>
<td>logical inference</td>
<td>70</td>
</tr>
<tr>
<td>minimization axiom</td>
<td>56, 176</td>
</tr>
<tr>
<td>Term</td>
<td>Page Numbers</td>
</tr>
<tr>
<td>---------------------------</td>
<td>--------------</td>
</tr>
<tr>
<td>mix inference</td>
<td>171</td>
</tr>
<tr>
<td>multistore</td>
<td>117</td>
</tr>
<tr>
<td>open formula</td>
<td>66</td>
</tr>
<tr>
<td>oracle</td>
<td>22</td>
</tr>
<tr>
<td>p-inductive definition</td>
<td>119, 125</td>
</tr>
<tr>
<td>p-inductive proof</td>
<td>123</td>
</tr>
<tr>
<td>parameter variables</td>
<td>75</td>
</tr>
<tr>
<td>pigeon-hole principle</td>
<td>102</td>
</tr>
<tr>
<td>polynomial growth rate</td>
<td>9, 22, 153</td>
</tr>
<tr>
<td>polynomial hierarchy</td>
<td>14</td>
</tr>
<tr>
<td>polynomial time closure</td>
<td>9, 23</td>
</tr>
<tr>
<td>polynomially bounded quantification</td>
<td>13</td>
</tr>
<tr>
<td>predicate</td>
<td>13</td>
</tr>
<tr>
<td>predicate oracle</td>
<td>22</td>
</tr>
<tr>
<td>predicative</td>
<td>210</td>
</tr>
<tr>
<td>principal abstract</td>
<td>164, 165</td>
</tr>
<tr>
<td>principal formula</td>
<td>70</td>
</tr>
<tr>
<td>proof</td>
<td>70, 132</td>
</tr>
<tr>
<td>proper</td>
<td>15</td>
</tr>
<tr>
<td>propositional inference</td>
<td>70</td>
</tr>
<tr>
<td>protosequences</td>
<td>44</td>
</tr>
<tr>
<td>quantifier inference</td>
<td>70</td>
</tr>
<tr>
<td>rank</td>
<td>172</td>
</tr>
<tr>
<td>relational</td>
<td>180</td>
</tr>
<tr>
<td>replacement axiom</td>
<td>53, 178</td>
</tr>
<tr>
<td>restricted by parameter variables</td>
<td>77</td>
</tr>
<tr>
<td>right rank</td>
<td>171</td>
</tr>
<tr>
<td>second order variable</td>
<td>160</td>
</tr>
<tr>
<td>semiformal</td>
<td>128</td>
</tr>
<tr>
<td>semiterm</td>
<td>127</td>
</tr>
<tr>
<td>sequent</td>
<td>67, 131</td>
</tr>
<tr>
<td>sharply bounded quantifier</td>
<td>20, 29, 160</td>
</tr>
<tr>
<td>son</td>
<td>117</td>
</tr>
<tr>
<td>space bound</td>
<td>8</td>
</tr>
<tr>
<td>structural inference</td>
<td>70</td>
</tr>
<tr>
<td>substitution instance</td>
<td>72, 170, 183</td>
</tr>
<tr>
<td>sucedent</td>
<td>67</td>
</tr>
<tr>
<td>successor</td>
<td>73</td>
</tr>
<tr>
<td>suitable polynomial</td>
<td>8</td>
</tr>
<tr>
<td>term</td>
<td>66, 127</td>
</tr>
<tr>
<td>time bound</td>
<td>8</td>
</tr>
<tr>
<td>tree</td>
<td>117</td>
</tr>
<tr>
<td>unbounded quantifier</td>
<td>20, 29, 160</td>
</tr>
<tr>
<td>uniform</td>
<td>23</td>
</tr>
<tr>
<td>weak free variable normal form</td>
<td>75</td>
</tr>
</tbody>
</table>