Self-Verifying Axiom Systems, the Incompleteness Theorem and Related Reflection Principles

Dan E. Willard *

Abstract

We will study several weak axiom systems that use the Subtraction and Division primitives (rather than Addition and Multiplication) to formally encode the theorems of Arithmetic. Provided such axiom systems do not recognize Multiplication as a total function, we will show that it is feasible for them to verify their Semantic Tableaux, Herbrand, and Cut-Free consistencies. If our axiom systems additionally do not recognize Addition as a total function, they will be capable of recognizing the consistency of their Hilbert-style deductive proofs. Our axiom systems will not be strong enough to recognize their Canonical Reflection principle, but they will be capable of recognizing an approximation of it, called the *"Tangibility Reflection Principle"*. We will also prove some new versions of the Second Incompleteness Theorem stating essentially that it is not possible to extend our exceptions to the Incompleteness Theorem much further.

NOTE TO THE READER: This article was published in the June 2001 of the *Journal of Symbolic Logic*, pp. 536-596. It interfaces naturally with our two more recent papers [44, 45]. These three papers can be read in any order that a reader desires.

*Address: Dep of CS, SUNYA, Albany, NY 12222 or dew@cs.albany.edu. Tel. 518-452-0148. Supported by NSF Grant CCR 99-02726

1. INTRODUCTION: Define an axiom system α to be **Self-Justifying** iff

i) α can formally verify its own consistency (by some reasonable definition of self-consistency), and

ii) the axiom system α is in fact consistent.

Rogers [30] has noted that Kleene's Fixed Point Theorem implies that every r.e. axiom system α can be easily extended into a broader system α^* which satisfies condition (i). Kleene's proposal was essentially for the system α^* to contain all α's axioms plus the one additional axiom sentence:

There is no proof of 0=1 from the union of α with *"THIS SENTENCE"*.

It is known [16, 30, 15]) that the Fixed Point Theorem makes it possible to encode the indented sentence above for any r.e. axiom system α. However, the catch is that α^* can be inconsistent even while its added axiom justifies α^* 's consistency. Thus α^* will typically violate Part-ii of the definition of Self-Justification. This problem arises not only in Gödel's paradigm, where α has at least the power of Peano Arithmetic (PA), but also for many systems much weaker than PA. For instance, no system satisfying the Hilbert-Bernays properties can be self-verifying [13, 14, 23]. Many other types of generalizations of Gödel's Second Incompleteness Theorem are also known [6, 8, 13, 28, 33, 34].

Let us define $x - y$ to equal zero when $x < y$. Similarly, say $\lfloor \frac{x}{y} \rfloor$ equals x when $y = 0$. Subtraction and Division, thus defined, are total functions. Let Φ denote some Π_1 sentence in the conventional language of Arithmetic which employs the usual function symbols for Addition and Multiplication. It is clearly possible to map each such Π_1 sentence Φ onto a sentence Ψ that replaces the Addition and Multiplication symbols with the Subtraction and Division function symbols and has the property that $\Phi \equiv \Psi$ in languages that are adequately strong. (This equivalence certainly does not hold in sufficiently weak languages.) The advantage of this approach is that it allows an axiom system to avoid the assumption that Addition and Multiplication are total functions. In the absence of such assumptions about the totality of Addition and Multiplication functions, our axioms systems will be certainly too weak to formally prove $\Phi \equiv \Psi$, but they will be able to work with a sentence Ψ that is in some sense a partial counterpart of the sentence Φ . Our theorems in Sections 3-5 will explain how this change in perspective allows an axiom system to avoid at least some of the limitations imposed by the Second Incompleteness Theorem.

The language that we will formally study is defined in section 2, and it will use seven function symbols for representing the operations of subtraction, division, predecessor, maximum, logarithm, root-finding, and counting the number of "1" bits in an integer's binary encoding. We will define Δ_0^- , Σ_1^- and Π_1^- sentences to be the exact analogs of conventional Arithmetic's Δ_0 , Σ_1 and Π_1 sentences except that our language will replace the Addition and Multiplication function symbols with these seven new alternate symbols. In this

context, $\mathrm{Prf}_\alpha(x, y)$ will denote a Δ_0^- formula indicating y is the Gödel number of a proof from axiom system α of the sentence with Gödel number x. (Some theorems describing how to encode $\mathrm{Prf}_\alpha(x, y)$ as a Δ_0^- formula will appear later in Appendices B through D, and the reader should temporarily just assume such an encoding is feasible.) If $\ulcorner \Phi \urcorner$ denotes Φ's Gödel number, then an axiom system α's **Canonical Reflection Principle** for the sentence Ψ is the sentence:

$$\{ \quad \exists\ y \quad \mathrm{Prf}_\alpha (\ulcorner \Psi \urcorner , \ y \ ) \quad \} \quad \supset \quad \Psi \tag{1}$$

Löb's Theorem [22] asserts *every consistent extension* α of Peano Arithmetic is capable of proving the validity of (1) in only the uninteresting degenerate case where α can prove Ψ by itself! Our research was motivated largely by the observation that it is possible for axiom systems α, not recognizing Addition and Multiplication as total functions, to justify close approximations to the impermissible reflection principle (1). They can thus evade much of what had appeared to be the implications of Löb's Theorem.

Our objectives are best summarized when it is noted that the following four generalizations of Gödel's Incompleteness Theorem severely limit the capacity of any self-justifying system:

A) Pudlák [28] has shown that no extension of Robinson's System Q can verify its own Hilbert consistency. That is, there can exist no consistent axiom system α which recognizes Addition and Multiplication as total functions and which can prove the non-existence of a Hilbert-style proof of 0=1 from itself. (See [7, 10, 13, 14, 23, 31] for the definition of a Hilbert proof.)

B) Robert Solovay (private communications [34]) has generalized Pudlák's theorem to systems α which merely recognize Successor as a total function and use Subtraction and Division to implicitly represent Addition and Multiplication. If such an α is consistent and can prove all Peano Arithmetic's Π_1^- theorems about Subtraction and Division, then Solovay's theorem asserts α cannot prove the non-existence of a Hilbert-proof of 0=1 from itself. (A detailed description of Solovay's theorem appears in Appendix A. In addition to Pudlák, Solovay credited research by Nelson and Wilkie-Paris [24, 38] for having influenced his generalization of Pudlák's theorem.)

C) A variation of Löb's Theorem is also valid for systems α which recognize none of Addition, Multiplication or even Successor as total functions. Our Theorem 7.2 will show that if such an α is consistent and can prove all Peano Arithmetic's Π_1^- theorems about Subtraction and Division then it cannot verify Equation (1)'s canonical reflection principle for every Π_1^- sentence Ψ.

D) Zofia Adamowicz has recently started circulating an unpublished manuscript [1] showing that for each $m \geq 2$, the Second Incompleteness Theorem can be generalized to show that $\mathrm{I}\Sigma_1 + \Omega_m$ is unable to prove a theorem verifying its own Semantic Tableaux consistency.

The perhaps most surprising aspect of our research is that it will provide several examples of interesting self-justifying systems, despite the limitations above. These systems will contain sufficient robustness to offer at least a reasonable approximation of the reflection principle (1), while also recognizing the validity of Peano Arithmetic's Π_1^- theorems about the properties of Subtraction and Division. We need two new definitions before we can explain how we will evade Löb's Theorem.

Definition 1.1. The literature has defined a sentence to be Prenex Normal iff it contains a block of quantifiers on its left side followed by an open formula. It is usually assumed that these quantifiers are unbounded, i.e. they are not quantifiers of the form $\forall x \leq t$ or $\exists x \leq t$ where t denotes an arbitrary term. We will say a sentence is written in **PRENEX*** form iff its quantifiers are permitted to be either bounded or unbounded quantifiers. Also if Φ denotes a Prenex* sentence, Φ_j^i will denote a sentence identical to Φ except that each unbounded universally quantified variable from Φ is bounded by i and each unbounded existentially quantified variable from Φ is bounded by j. (Bounded quantifiers do not have their range change.) The symbol Φ^i will be an abbreviation for Φ_∞^i .

Definition 1.2. Let $\text{Tangible}(x)$ be a formula which asserts that the number x is not of unusually large size, i.e. its size is very "tangible". Since our axiom systems α will not necessarily assume that any of Multiplication, Addition or even Successor are total functions, three possible definitions of tangibility are given below. (In these definitions, it is assumed that the constant $k \geq 2$.)

a. $\text{TangPred}\,(x) \quad = \quad \{ \ \exists v \ \ x < v - 1 \ \}$

b. $\text{TangDiv}_k\,(x) \quad = \quad \{ \ \exists v \ \ x < \frac{v}{k} \ \}$

c. $\text{TangRoot}_k\,(x) \quad = \quad \{ \ \exists v \ \ x < v^{1/k} \ \}$

In a notation where $\text{Tangible}(x)$ denotes any one of $\text{TangPred}(x)$, $\text{TangDiv}_k(x)$ or $\text{TangRoot}_k(x)$, an axiom system α's **Tangibility Reflection Principle** for the sentence Ψ is defined to be the assertion:

$$\forall x \quad \{ \quad [\quad \exists \ y \quad \text{Prf}_{\,\alpha} \ (\ \ulcorner \Psi \urcorner \ , \ y \ ) \ \wedge \ \text{Tangible}(x) \quad] \quad \supset \Psi^x \quad \} \tag{2}$$

Our main theorems will define axiom systems α which can verify their tangibility reflection principles, for all prenex* sentences Ψ simultaneously. They will thus establish the following results:

i. Assume that "Prf" in Equation (2) denotes either a Hilbert proof method or Gentzen's Sequent Calculus (with deductive cuts permitted). For each consistent arithmetic axiom system A , there will exist an axiom system α which supports the TangPred version of the tangibility reflection principle and can also verify all A's Π_1^- theorems (when these theorems are rewritten in a form where the Subtraction and Division primitives replace Addition and Multiplication). These systems α will not recognize

either Addition or Multiplication as total functions. However, they can recognize Bitwise-Or as a total function when the predicate TangDiv_2 replaces TangPred in Equation (2).

ii. The systems α (above) can be modified to recognize Addition as a total function when "Prf" in Equation (2) is changed to designate some cut-free formalism (such as Herbrand or Semantic Tableaux proofs) and when the TangRoot version of the tangibility reflection principle is employed.

One of the curious aspects of Items (i) and (ii) above is the extremely tight fit between their positive results and the four negative results (A) through (D), mentioned earlier. For example, Solovay's version of the Incompleteness Theorem (stated in B) explains why α in (i) or (ii) could not possibly simultaneously verify some version of its Hilbert consistency and also recognize successor as a total function. Similarly, Item C explains why it is necessary for Equation (2) to employ some version of a tangibility predicate (since the unmodified canonical reflection principle (1) is always infeasible).

In order to compare our results more closely to the prior literature, it is helpful to introduce one further definition. Say a formula $\Upsilon(v)$ is a **Definable Cut** for an an axiom system α if α can prove

a. $\Upsilon(k)$ for each fixed natural number k,

b. $\forall u \forall v \;\; \Upsilon(v) \wedge u < v \;\supset\; \Upsilon(u)$, and

c. $\forall u \;\; \Upsilon(u) \;\supset\; \Upsilon(u+1)$.

Also, if α can prove only the first two of the above three conditions about $\Upsilon(v)$, then $\Upsilon(v)$ will be said to satisfy α's weaker **"Tangibility Criteria"**. The prior literature [20, 21, 24, 28, 36] has illustrated several examples of different formulae $\text{Tang}(v)$, which are Definable Cuts for α, such that α can prove its consistency local to these Cuts. Thus, it has shown how several quite different axiom systems α can prove:

$$\forall y \quad \{ \quad \text{Tang}(y) \quad \supset \quad \neg \, \text{Prf}_{\alpha} \, (\, \ulcorner \, 0 = 1 \, \urcorner \, , \, y \,) \quad \} \tag{3}$$

One example of such a system was due to Kreisel and Takeuti [20, 21, 36]. They studied a type of self-justifying axiom system, based on a Second-Order Logic formalism that is a generalization of Gentzen's Cut-Free Sequent Calculus [11]. Their systems could prove the statement (3) about themselves when $\text{Tang}(y)$ was taken to be a second-order logic definition of the Natural Numbers and "Prf" denoted a second-order generalization of Gentzen's Cut-Free Sequent Calculus. Nelson [24] proved that Robinson's System Q (with linear ordering) could prove a version of (3) about itself when Tang(y) was taken to be a delicately specified "Definable Cut" of the Natural Numbers and "Prf" denoted Herbrand deduction. Pudlák [13, 28] proved a more general theorem for any finitely axiomatized sequential theory (and also allowing for the Wilkie-Paris notion [38] of a Herbrand-restricted-consistency proof). Examples of axiom systems that can prove their consistency on Definable Cuts thus include $\text{I}\Sigma_0$, $\text{I}\Sigma_0$+Exp, ACA_0, and GB, among many others.

A slightly different perspective on an axiom's ability to partially verify its own consistency is seen in Wilkie-Paris [27, 38]'s examination of $I\Sigma_0$ and its generalizations. Six theorems are discussed in [38]. Among these are that $I\Sigma_0 + \Omega_1$ can prove that its consistency is equivalent to the consistency of both $I\Sigma_0$ and $I\Sigma_0 + \Omega_n$ (for any $n > 0$), and that $I\Sigma_0$+Exp is incapable of proving the consistency of $I\Sigma_0$ (Indeed, it cannot prove the consistency of even Robinson's System Q.) Yet at the same time, it is known that added growth power of *SuperExp* would enable $I\Sigma_0$+*SuperExp* to prove $I\Sigma_0$'s consistency. In combination, these facts indicate $I\Sigma_0$ has a partial ability to appreciate its own self-consistency (since it can appreciate that the hypothesis that SuperExp is a total function implies its consistency). On the other hand, Wilkie-Paris showed that there is a well-defined limit to $I\Sigma_0$'s understanding of its own self-consistency (because $I\Sigma_0$+*Exp* cannot prove the consistency of $I\Sigma_0$).

Our research is partially related to the prior research of Kreisel, Nelson, Paris Pudlák, Takeuti and Wilkie because both the Equations (2) and (3) employ some versions of tangibility formula. However, it is difficult to make more detailed comparisons because each approach offers differently advantageous insights. Roughly speaking, the advantages of the perspectives of [20, 21, 24, 28, 36, 38] are that:

I. Many of the axiom systems of [20, 21, 24, 28, 36, 38] were widely studied axiom systems, such as Robinson's Q, ACA_0, $I\Sigma_0$, GB, and Gentzen Systems, unlike our systems which were especially manufactured with the assistance of Kleene's Fixed Point Principle.

II. Each of the systems in [20, 21, 24, 28, 36, 38] recognized Multiplication as a total function, unlike our systems, which in one case (the system ISREF(A)) will fail to view Successor as a total function.

III. As a consequence of Item (II), ISREF(A) will employ Equation (2)'s reflection principle in a context where it fails to view TangPred(x) as a Definable Cut. In contrast, each of [20, 21, 24, 28, 36, 38] will recognize their Tang(y) formulae in Equation (3) as Definable Cuts. (Also unlike ISREF(A), Section 6's $IS^{\lambda}(A)$ system will recognize its comparable TangRoot(x) formula as a Definable Cut.)

IV. $I\Sigma_0$ can prove that if *SuperExp* is a total function then it is consistent [13, 38]

On the other hand, the new results proven in Sections 3-6 are that:

V. The variable y in Equation (2) will *not* need to be required to satisfy any form of restricting predicate Tang(y). (Only x is so restricted.) In particular, Sections 3-6 will formalize axiom systems α, which differ from Equation (3)'s α by being capable of proving: $\forall y \neg \mathrm{Prf}_{\alpha}(\ulcorner 0 = 1 \urcorner , y)$.

VI. The reflection principle (2) is an inherently broader statement than Equation (3)'s mere assertion of the non-existence of a proof of 0=1, and

VII. The version (i) of our result, unlike previous efforts at self-verification, permits "Prf" to designate either an Hilbert-proof system or a Gentzen-system allowing deductive cuts.

It is infeasible to devise a *hybridized* self-justifying axiom formalism, that simply combines our methods with the prior literature [20, 21, 24, 28, 36, 38] because such a formalism would violate one of the four variants, (A) through (D) of the Incompleteness Theorem, mentioned earlier. Hence, there are inherent tradeoffs that prevent a self-justifying axiom system from possessing all of features (I) through (VII), *simultaneously!* Thus, it is futile to seek an idealized form of self-verifying system, that combines the advantages of the many different approaches *simultaneously.* Each should be viewed as providing differently desirable insights.

It should be kept in mind that Feferman's discussion of axiom systems that evade Gödel's Incompleteness Theorem by capturing the "numerical" but "not intensional" definitions of provability is certainly relevant to both our work and to the prior literature [20, 21, 24, 28, 36, 38] because it recognizes that certain types of exceptions to the Incompleteness Theorem are feasible when one sufficiently stretches and modifies the notion of provability. However, it does not anticipate the precise results that we or others derived because it does not discuss which precise directions the notion of provability can be maximally stretched.

FURTHER COMMENTS ABOUT THE PRIOR LITERATURE: Numerous articles [13, 24, 26, 28, 29, 38] have credited an unpublished observation by Solovay for introducing the method of "thinning" a Definable Cut. The development of cut formulae for Robinson's System Q (that can model $I\Sigma_0 + \Omega_n$) were due to both Nelson [24] and Wilkie-Paris[38, 39]. The proofs of the (A) and (B) versions of the Incompleteness Theorem, by Pudlák and Solovay [28, 34], employs such models of $I\Sigma_0 + \Omega_n$, together with the thinning of cut formulae. The thinning method was also employed by Nelson and Pudlák [24, 28] to establish that Robinson's Q and many of its extensions can prove Equation (3)'s self-consistency statement for Herbrand Deduction. It was also used by Paris and Wilkie in their study of $I\Sigma_0$ and its generalizations.

Buss, Dimitracopoulos, Hájek, Krajícek, Parikh, Paris, Pudlák and Wilkie in [6, 12, 13, 17, 18, 25, 26, 29, 38] have discussed many other respects how Definable Cuts and/or fastly growing functions affect a system's ability to recognize its self-consistency. It is known that no axiom system stronger than $I\Sigma_0 + Exp$ can prove its Herbrand consistency. In this connection surveying both his work and that of Paris-Wilkie [27], Pudlák explicitly states on page 435 of [28] that it is unknown what systems weaker than $I\Sigma_0 + Exp$ will have a similar property. Our $IS(A)$ and $IS^{\lambda}(A)$ axiom system, in Section 4-6, provide a partial answer to this question because they verify their Semantic Tableaux (and also Herbrand) consistency while recognizing Addition as a total function and also proving Peano Arithmetic's Π_1^- theorems.

OTHER INTRODUCTORY COMMENTS: Since none of our axiom systems will recognize Multiplication as a total function and since some will also not recognize Addition, these axiom systems are clearly awkward in some serious respects. The reason for our interest in such awkwardly defined axiom systems is that they offer a partial answer to a version of the Liar's Paradox raised by Gödel's Incompleteness Theorem. In particular, they will provide a possible partial answer to the paradoxical question below:

* How do Human Beings manage to muster the physical energy and psychological desire to think (and prove theorems) when the many generalizations of Gödel's Incompleteness Theorem assert that no reasonable conventional axiom system can confidently assume its own consistency?

We can provide no perfect answer to the preceding question because there can obviously never exist any completely satisfactory answer to a logical paradox. However, our partial answer to the logical paradox $*$, raised by Gödel's Incompleteness Theorem, will be that a Thinking Being can indeed assume that if he proves Ψ, then Ψ will be valid when it is restricted to all numbers of at least reasonable size (i.e. the "tangible" numbers specified in Equation (2)).

Part of what will make the Reflection Principle (2) interesting is that the Variation-C of the Incompleteness Theorem shall demonstrate that Equation (1)'s slightly more general canonical reflection principle *always* leads to a *hopelessly inconsistent axiom system.* Thus since (1) is **always infeasible**, it is curious that the compromise reflection principle (2) is often *permissible.*

The Tangibility Reflection Principle is especially curious when Ψ in Equation (2) represents what the next section shall call "a Σ_1^- sentence". (Section 2 defines Σ_1^- sentence to be roughly the analog of conventional arithmetic's Σ_1 sentences where the Addition and Multiplication function symbols are now replaced with Section 2's seven function symbols of Subtraction, Division, Logarithm ...) The crucial aspect of any Σ_1^- sentence Ψ is that $\Psi \equiv \Psi^0$, and that essentially any axiom system α can verify this equivalence. Therefore for any Σ_1^- sentence Ψ, our self-verifying systems α will be able to further deduce from Ψ's Tangibility axiom (2) the following inference about its own self-consistency:

$$\{ \quad \exists \ y \quad \mathrm{Prf}_{\alpha} (\ulcorner \Psi \urcorner , \ y \ ) \ \} \quad \supset \quad \Psi \tag{4}$$

Moreover, if $\Psi \equiv$ " 0=1 " then α can further deduce Equation (5) from (4).

$$\forall y \quad \neg \ \mathrm{Prf}_{\alpha} (\ulcorner 0 = 1 \urcorner , \ y \ ) \tag{5}$$

The above two inferences from (2)'s Tangibility Reflection Principle illustrate how α can deduce some very conventionally stated formulations of its self-consistency from the tangibility principle.

We wish to close this section by again acknowledging that there are disadvantages to our self-justifying axiom systems, since the presence of the Tangibility Reflection Principles (2) will force all our axiom systems

to drop the assumption that Multiplication is a total function. Also, it is less than ideal to remove the axiom that Addition is a total function (as one of our three axiom systems will do). However, paradoxes never have perfect solutions. The partial virtues of the IS$^{\lambda}(A)$ and ISREF(A) axiom systems is that their Tangibility Reflection Principles will offer at least a partial answer to the Paradoxical Question $*$.

2 The Definitions of IS(A), ISREF(A) and IS$^{\lambda}(A)$

The acronym "IS" stands for "Introspective Semantics". Let us assume that the "base axiom system" A is an extension of Peano Arithmetic. Our "IS(A)" axiom system will be designed to prove all the Π_1^- arithmetic theorems of A plus additionally contain an ability to verify that IS(A), *itself*, does not contain a proof of 0=1. This section will define the axiom system IS(A) and some of its generalizations.

Define $F(a_1, a_2, ...a_j)$ to be a **NON-GROWTH FUNCTION** iff for all values of $a_1, a_2, ...a_j$, the function F satisfies $F(a_1, a_2, ...a_j) \leq Maximum(a_1, a_2, ...a_j)$. Our axiom systems will employ a set of seven non-growth functions, called the **GROUNDING FUNCTIONS**. They will include:

1. Integer Subtraction where $x - y$ is defined to equal zero when $x < y$,
2. Integer Division where $\frac{x}{y} = x$ when $y = 0$, and it otherwise equals $\lfloor \frac{x}{y} \rfloor$.
3. $Predecessor(x) = \text{Max}(x - 1, 0)$,
4. $Maximum(x, y)$,
5. $Logarithm(x) = \lceil Log_2(x + 1) \rceil$,
6. $Root(x, y) = \lfloor x^{1/y} \rfloor$ when $y \geq 1$, and $Root(x, 0) = x$.
7. $Count(x, j)$ designating the number of "1" bits among x's rightmost j bits.

We will follow mostly conventional logic notation when discussing the functions above. Thus a *term* is defined to be a constant symbol, a variable symbol or a function symbol (followed by some input arguments, which are similarly defined terms). If t is a term then the quantifiers in $\forall v \leq t \ \Psi(v)$ and $\exists v \leq t \ \Psi(v)$ will be called *bounded quantifiers*. These two wffs will be semantically equivalent to the formulae $\forall v \ (v \leq t \supset \Psi(v))$ and $\exists v \ (v \leq t \wedge \Psi(v))$. A formula Φ will be called Δ_0^- iff all its quantifiers are bounded. A sentence will be called Π_1^- (respectively Σ_1^-) iff for some Δ_0^- formula $\Phi(v_1, v_2, ...v_n)$, it is written in the form $\forall v_1 \ \forall v_2 \ ... \ \forall v_n \ \Phi(v_1, v_2, ...v_n)$, (respectively $\exists v_1 \ \exists v_2 \ ... \ \exists v_n \ \Phi(v_1, v_2, ...v_n)$).

We will now define the **"IS(•) MAPPING"** . It will map an initial axiom system A onto an axiom system IS(A), such that the latter can simultaneously recognize that there exists no proof of 0=1 from IS(A) and prove all of A's Π_1^- theorems. In particular, IS(A) will consist of the following four groups of axioms:

Group-Zero: The axiom system IS(A) will contain one constant symbol $\bar{n}$ for each natural number $n \geq 0$. The Group-Zero axioms will define these constants formally. They will include $\bar{1} \neq \bar{0}$, and for each $n > 0$, the axioms: Predecessor($\overline{n}$) $= \overline{n-1}$, $\overline{2n} - \overline{n} = \overline{n}$ and $\overline{2n+1} - \overline{n} - \bar{1} = \overline{n}$.

Group-1: The Π_2^- axiom (below) indicates that Addition is a total function:

$$\forall x \; \forall y \; \exists z \quad z - x = y \tag{6}$$

The other axioms in Group-1 will be a set of Π_1^- sentences assigning the "=" and "<" predicates their usual logical properties and assuring that for each Grounding Function F and for each set of constants $\bar{k}$, $\bar{c_1}$, $\bar{c_2}, ...\bar{c_m}$, the Group-Zero and Group-1 axioms will together imply $F(\bar{c_1}, \bar{c_2}, ...\bar{c_m}) = \bar{k}$ whenever this sentence is true. Table I (at the end of this article) illustrates one such finite set of Π_1^- axiom sentences, but any other finite set of Π_1^- axioms with these properties is equally suitable. Our theorems will also be valid if Group-1 is expanded to include any larger set $\mathcal{F}$ of additional non-growth functions that is axiomatized by any finite set of Π_1^- axioms (and the Group-2 scheme is accordingly adjusted).

Group-2: The Group-2 axiom scheme will provide IS(A) with the ability to prove all A's Π_1^- theorems. More formally, let T denote a transformation which maps each Π_1^- sentence Φ (in IS(A)'s language) onto its counterpart in A's language (which is denoted as Φ^T). Let TransProof$_A$(x, y) be a Δ_0 formula indicating x is the Gödel number of some sentence in IS(A)'s language and y is a proof from axiom system A of a theorem Φ^T (where Φ denotes the sentence specified by x). Then for each Π_1^- sentence Φ , the Group-2 schema will contain a corresponding axiom:

$$\forall \, y \quad \{ \text{ TransProof }_A (\ulcorner \Phi \urcorner , \, y) \quad \supset \quad \Phi \quad \} \tag{7}$$

The axiom (7) will be encoded as a Π_1^- sentence. The footnote [1] explains why we use Equation (7) to formulate the Group-2 axioms (rather than simply list A's Π_1^- theorems as axioms). The footnote [2] explains how the Group-2 translation function T has connections to the literature on Definable Cuts.

[1] We will desire the Group-2 axiom schema to be defined by a Δ_0^- formula, called Ax(s), which checks to see if a sentence s is an axiom. Most axiom systems A would not allow Ax(s) to be a Δ_0^- formula, if one automatically converted all A's theorems into proper axioms. On the other hand, a Δ_0^- formula can recognize every sentence s whose generic form is described by Equation (7). Thus, the generic form, described by Equation (7), is desirable because IS(A) will then have access to a Δ_0^- formula for recognizing its Group-2 axioms.

[2] This footnote is not essential to an understanding of our paper, but readers familiar with the literature on on Definable Cuts [12, 13, 17, 18, 24, 26, 28, 29, 38] may appreciate it. Let Φ^J be the modification of the sentence Φ that is relativized with respect to the definable cut formula J . Then for each different Definable Cut formula J , one can define a distinctly different mapping T_J which maps Φ onto Φ^J . For some axiom systems, such as Peano Arithmetic or ZF-Set Theory, there is only one available definable cut. For other axiom systems, such as for example GB-Set Theory or IΣ_0, there exists an infinite number of different eligible definable cuts and corresponding formulations of Equation (7)'s mapping function T_J . For simplicity, we will assume that one such mapping function T is specified in advance, and it will be the only mapping function employed by IS(A). The theory of Definable Cuts would allow us to consider some more complex situations, but the presentation in this paper will be far simpler if we just put aside such added considerations and simply assume that one straightforward mapping function T is specified in advance.

Group-3: Let $\mathrm{SemPrf}_{\alpha}(x, y)$ be a Δ_0 formula indicating y is the Gödel number of a Semantic Tableaux proof from axiom system α of the sentence with Gödel number x. (The definition of Semantic Tableaux proofs can be found in [10, 23, 32], and it is also reviewed by us in Section 4). The Group-3 axiom of IS(A) will consist of a single sentence stating that there exists no Semantic Tableaux proof of 0=1 from IS(A). This Group-3 axiom will be equivalent to Equation (8), encoded as a Π_1^- sentence:

$$\forall y \quad \neg\, \mathrm{SemPrf}_{\mathrm{IS(A)}}\, (\, \ulcorner\, 0 = 1 \,\urcorner\, ,\ y\ ) \tag{8}$$

Because the axiom (8) refers to an axiom system which includes itself, its formal encoding (in Appendix B) involves a diagonalization construction, employing the Fixed Point Theorem. Our article has been written so that the reader accepting the claim that the Fixed Point Theorem makes it possible to formally encode the Axiom (8), can understand all the main theorems without viewing Appendix B's detailed encoding of it. (Two other axiom systems, besides IS(A) are also discussed in Appendix B. Therefore, the reader should postpone examining Appendix B until after he completes this section.)

Some readers may be initially surprised that IS(A) is able to recognize its self-consistency simply by incorporating the Group-3 statement, asserting its consistency, as a formal axiom. It would be natural for many readers to thereby wonder: *"How is IS(A) nontrivial?"* The answer was provided in the first paragraph of Section 1. It noted that every r.e. axiom system can be easily extended to satisfy Part (i) of the definition of self-justification (by applying Kleene's Fixed Point Theorem). However, *the catch was* that most such extended axiom systems fail "*Part (ii)*" of this definition. That is, most axiom systems become inconsistent when they are augmented to include an axiom simply declaring that "*I am consistent*". However, IS(A) will be different. Theorem 4.3 will prove that *if A is consistent then IS(A) is automatically consistent.* Thus, if A = Peano Arithmetic, IS(A) can prove all the Π_1^- theorems of Peano Arithmetic, and its consistency will be as certain as that of Peano Arithmetic !

Below are defined two generalizations of IS(A), called IS$^{\lambda}$(A) and ISREF(A), which we will discuss:

IS$^{\lambda}$(A) will assume λ denotes a fixed constant satisfying $0.01 < \lambda < 1$. Its Group-zero, Group-1 and Group-2 axioms are identical to those of IS(A). However, its Group-3 schema will be strictly more powerful than IS(A)'s counterpart. Let Ψ denote a Prenex* sentence. Let Definition 1.1's notation define Ψ_z^x . For each Prenex* sentence Ψ, the Group-3 schema of IS$^{\lambda}$(A) will contain a corresponding " $\lambda-$reflection axiom" which is equivalent to Equation (9) encoded as a Π_1^- sentence:

$$\forall x\, \forall y\, \forall z \quad \{ \quad \mathrm{SemPrf}_{\mathrm{IS}^{\lambda}(A)}\, (\, \ulcorner\, \Psi \,\urcorner\, ,\ y\ ) \ \wedge\ y^{\lambda} < \frac{z}{x} \quad \supset \quad \Psi_z^x \ \} \tag{9}$$

The formal encoding of IS$^{\lambda}$(A)'s Group-3 axiom appears in Appendix B. It uses a Fixed Point construction analogous to the encoding of IS(A)'s Group-3 axiom.

ISREF(A) will employ the same Group-zero and Group-2 schemas as IS(A) and IS$^\lambda$(A). Its Group-1 schema, however, will be weaker by omitting the axiom that Addition is a total function (i.e. Equation (6) is omitted). The advantage of this omission is that it will allow ISREF(A) to use a much stronger Group-3 schema, employing Hilbert rather than Semantic Tableaux deduction. In particular, let $\text{HilbPrf}_\alpha(x, y)$ denote a formula indicating y is the Gödel number of a Hilbert-proof from α of the theorem x. Define **SIZE(y)** to be a function where $\text{Size}(y) = c$ when y is the Gödel number of a proof whose largest stored constant is c. Then for each Prenex* sentence Ψ, ISREF(A) will contain a corresponding Π_1^- axiom (encoded in Appendix B) which is equivalent to the sentence:

$$\forall x\ \forall y\ \ \{\text{HilbPrf}_{\text{ISREF}(A)}(\ \ulcorner \Psi \urcorner\ ,\ y\ )\ \ \wedge\ \ \text{Size}(y) \leq x - 1\ \ \supset\ \ \Psi_{x-1}^{x-1}\ \ \} \tag{10}$$

One reason the axioms (9) and (10) are interesting is that they will enable IS$^\lambda$(A) and ISREF(A) to verify their respective TangRoot and TangPred reflection principles (i.e. see Theorems 3.5 and 6.2). Moreover, Equations (4) and (5) had shown how these Tangibility Principles permit an axiom system α to verify " $\forall y\ \neg\ \text{Prf}_\alpha(\ulcorner 0 \neq 1 \urcorner, y)$ " and also to recognize its Σ_1^- Reflection Principle.

Say an axiom system A is **Regularly Consistent** iff $\text{Prf}_A(x, y)$ has a Δ_0^- encoding and the generic Group-2 axioms (of Equation (7)) are valid for every Π_1^- sentence Φ. Define the "mapping" I($\bullet$) to be **Consistency-Preserving** iff I(A) is consistent whenever A is regularly consistent. Sections 3, 4 and 5 will prove that the three mappings IS($\bullet$), ISREF($\bullet$) and IS$^\lambda$($\bullet$) are each consistency-preserving. Sections 3 and 6 will prove that ISREF(A) and IS$^\lambda$(A) can verify their TangPred and TangRoot reflection principles. Section 7 will prove three new versions of the Second Incompleteness Theorem, showing that ISREF(A) and IS$^\lambda$(A) cannot be improved much further. For instance, one of Section 7's theorems will show that no reasonable axiom can support Equation (1)'s Canonical Reflection Principle.

This article will also include several Appendices. The Appendix A briefly summarizes a very pretty unpublished theorem by Robert Solovay. It oddly implies all the seemingly natural hybrids of ISREF(A) and IS(A) are actually infeasible. (Appendix A also describes related research of Nelson, Paris, Pudlák and Wilkie [24, 28, 38], which Solovay credited for partially inspiring his theorem-proof.) Applications of the Fixed Point Theorem that provide a formal Gödel-encoding for the Group-3 axioms of IS(A), ISREF(A) and IS$^\lambda$(A) are illustrated in Appendices B through D (with different levels of mathematical terseness). The paper has been organized so that the reader can examine the Appendices either before or after Sections 3-7.

3 Analysis of ISREF(A)

Our discussion will begin with ISREF(A) because it is easy to analyze. We first need two lemmas:

Lemma 3.1 Let Φ denote a prenex* sentence. Then Definition 1.1's notation implies:

A) $a \leq b \wedge \Phi_a^i \supset \Phi_b^i$

B) $a \geq b \wedge \Phi_j^a \supset \Phi_j^b$

C) $\Phi_j^i \supset \Phi^i$

D) $a \geq b \wedge \Phi^a \supset \Phi^b$

Proof. Immediate from Φ_j^i's definition (given in Definition 1.1). $\square$

Lemma 3.2. Let M_i denote the finite model of the natural numbers which assumes that only the integers $0, 1, 2, ...i$ exist. Let Θ denote a Π_1^- sentence whose constant symbols represent numbers $\leq i$. The validity of Θ in the Standard Model of the Natural Numbers implies Θ is also valid in the model M_i.

Proof Summary. The key point is that Π_1^- sentences are built out of Grounding Functions (which satisfy a non-growth property), rather than out of the more conventional *growth-oriented functions* of Addition and Multiplication. This *non-growth* property makes it trivially obvious Lemma 3.2 is valid. $\square$

Remark 3.3. If Θ is a Σ_1^- or Δ_0^- sentence whose constants are $\leq i$, the converse of Lemma 3.2 will be valid, i.e. the validity of Θ in M_i will imply its validity in the Standard Model of the Natural Numbers.

Theorem 3.4. The regular consistency of A implies that ISREF(A) is consistent. (In other word, ISREF($\bullet$) is a "Consistency-Preserving" mapping.)

Proof-by-Contradiction. Let us suppose the contrary, and assume ISREF(A) is inconsistent but A is regularly consistent. The latter certainly implies ISREF(A)'s Group-zero, Group-1 and Group-2 axioms are all valid in the Standard Model of the Natural Numbers. Hence, some Group-3 axiom (similar to Equation (10)) is invalid in the Standard Model (because otherwise ISREF(A) would be consistent). Thus, there must exist a triple (Ψ, p, i) satisfying

$$** \qquad \text{HilbPrf}_{\text{ISREF}(A)} (\ulcorner \Psi \urcorner , p) \quad \text{and} \quad i \geq \text{Size}(p) \quad \text{and} \quad \neg \Psi_i^i$$

Therefore, let (Ψ, p, i) denote the triple satisfying $**$ with *minimal value* in its third component. Our proof will construct another triple (Φ, q, j) with $j < i$ to establish the desired contradiction.

Our construction will begin with the observation that the combination of $**$ and Lemma 3.2 trivially implies (see footnote[3]) that Ψ is invalid in the finite model M_i. Since p employs axioms from ISREF(A)

[3]The statement $**$ indicates $\neg\Psi_i^i$ is valid in the Standard Model of the Natural Numbers and its stored constants are bounded by i. Also, $\neg\Psi_i^i$ is a Π_1^- sentence (indeed, it is Δ_0^- because of its superscript and subscript). Thus Lemma 3.2 implies $\neg\Psi_i^i$ is valid in the finite model M_i. Since $\Psi \equiv \Psi_i^i$ *within the model* M_i, we conclude that $\neg\Psi$ is also valid in the finite model M_i.

to prove a theorem Ψ (which is invalid in M_i), certainly *some axiom appearing in* p must be invalid in M_i . The footnote [4] explains how Lemma 3.2 immediately implies that all the Group-Zero, Group-1 and Group-2 axioms in p are valid in M_i . Thus, since all other possibilities are precluded, some Group-3 axiom lying in p must be invalid in M_i . The generic form of ISREF(A)'s Group-3 axioms is:

$$\forall x \, \forall y \quad \{ \quad \text{HilbPrf}_{\text{ISREF}(A)} (\ulcorner \Phi \urcorner , \, y) \quad \wedge \quad \text{Size}(y) \leq x - 1 \quad \supset \quad \Phi_{x-1}^{x-1} \quad \} \tag{11}$$

Since (11) is invalid in M_i , there will exist a triple (Φ, q, j) in the model M_i satisfying:

$$\text{HilbPrf}_{\text{ISREF}(A)} (\ulcorner \Phi \urcorner , \, q) \quad \wedge \quad j \geq \text{Size}(q) \quad \wedge \quad \neg \, \Phi_j^j \quad \wedge \quad j < i \tag{12}$$

Appendix B and C's encoding methods will imply that the formula " $\text{HilbPrf}_{\text{ISREF}(A)}(\ulcorner \Phi \urcorner, q)$ " in (12) (as well as its other three formulae) are Δ_0^- encodable. Thus, Equation (12) is a valid Δ_0^- sentence in the model M_i , and (by Remark 3.3) it will consequently also be valid in the Standard Model of the Natural Numbers. The triple (Φ, q, j) thus satisfies $**$ and contradicts the minimality of (Ψ, p, i) because $j < i$. Hence, ISREF(A) must be consistent to avoid this contradiction. □

General Comment: The proofs that IS(•) and IS^{λ}(•) are consistency-preserving are very different from Theorem 3.4's proof, but one aspect of them will be similar. They will be proofs by contradiction, which begin with a minimal element hypothetically violating the desired theorem. They will then use this minimal element to derive a contradiction by constructing a yet smaller element.

Theorem 3.5. Assume ISREF(A) employs an encoding scheme satisfying Equation (13). (It assures that all constants within any proof y are smaller than $y - 1$. Most encoding schemes, including Appendix B's method, satisfy (13) simply because they encode constants as binary numbers.) Then ISREF(A) can prove the validity of its TangPred Reflection Principle for any prenex* sentence Ψ .

$$\forall \, y \quad \{ \quad \text{Size}(y) \quad < \quad y - 1 \quad \} \tag{13}$$

Proof. Since ISREF(A) can prove Lemma 3.1.C, it can verify $\Psi_{x-1}^{x-1} \supset \Psi^{x-1}$. Then from its Group-3 axiom (in Equation (10)), ISREF(A) can immediately infer

$$\forall x \, \forall y \quad \{\text{HilbPrf}_{\text{ISREF}(A)} (\ulcorner \Psi \urcorner , \, y) \quad \wedge \quad \text{Size}(y) \leq x - 1 \quad \supset \quad \Psi^{x-1} \quad \} \tag{14}$$

From (13) and (14) ISREF(A) can obviously deduce:

$$\forall \, y \quad \{ \quad \text{HilbPrf}_{\text{ISREF}(A)} (\ulcorner \Psi \urcorner , \, y) \quad \supset \quad \Psi^{y-1} \quad \} \tag{15}$$

[4] We already noted that each Group-Zero, Group-1 and Group-2 axiom is a Π_1^- sentence, which is valid in the Standard Model. Since $**$ indicates $i \geq \text{Size}(p)$, Lemma 3.2 trivially implies each such axiom appearing in the proof p is valid in M_i .

Using Lemma 3.1.D, ISREF(A) can verify $[\, x \leq y \wedge \Psi^{y-1} \,] \supset \Psi^{x-1}$. Then from this identity and (13) through (15), ISREF(A) can trivially verify " $\forall x \forall y \, \{ \, \text{HilbPrf}_{\text{ISREF}(A)}(\ulcorner \Psi \urcorner, y) \supset \Psi^{x-1} \, \}$ ". The latter is equivalent to ISREF(A)'s TangPred Reflection Principle (given in Equation (2)). $\Box$.

Remark 3.6. One reason Theorems 3.4 and 3.5 are surprising is that the complementary negative results in Theorem 7.2 will imply essentially that *no conceivable* consistent axiom *of any type* can prove the analog of (15) for every sentence Ψ with the superscript $y-1$ changed to y .

Remark 3.7. Recall ISREF(A)'s Group-1 axioms do not recognize Successor as a total function (unlike IS(A)'s counterparts). It is fascinating that the Pudlák and Solovay versions of the Second Incompleteness Theorem imply that *even if* ISREF(A)'s Group-3 axioms were *sharply weakened* to state merely "I cannot produce a *Hilbert-Proof* of 0=1", this modified-ISREF(A) would *still be inconsistent* if it merely recognized Successor as a total function ! (See Appendix A for more details.)

Remark 3.8 Define ISTR(A) to be a modification of ISREF(A) whose Group-1 schema can recognize Bitwise-Or as a total function and whose Group-3 Scheme has the form:

$$\forall x \, \forall y \, \forall z \;\; \{ \;\; [\, \text{HilbPrf}_{\text{ISTR}(A)} \, (\, \ulcorner \Psi \urcorner \, , \; y \,) \;\; \wedge \;\; \text{Size}(y) < x \leq \frac{z}{2} \,] \;\; \supset \;\; \Psi^x_z \;\; \} \tag{16}$$

The analogs of Theorems 3.4 and 3.5 in Reference [43] show ISTR($\bullet$) is a Consistency-Preserving Mapping and that ISTR(A) can prove the TangDiv$_2$ versions of its Tangibility Reflection Principles. The TangDiv$_2$ Reflection Principles are of course weaker than ISREF's TangPred Principles. However, the interesting aspect of ISTR is that its Bitwise-Or function formalizes finite-set union (when integers represent finite sets as Bit-vectors). Moreover, the Group-2 axioms recognize bitwise set-subtraction and set-intersection operations (when A =Peano Arithmetic). Thus, ISTR(A) recognizes a TangDiv$_2$ formalization of *its Hilbert consistency, the totality of the three basic finite-set operations, and the validity of Peano Arithmetic's* Π_1^- *theorems.* Could this be a partial (albeit not complete) explanation for how Human Beings seem to have an instinctive sense of their Self-Consistency despite the limitations of the Second Incompleteness Theorem?

In the interests of brevity, we will not discuss ISTR(A) any further. (The proof that it satisfies analogs of Theorems 3.4 and 3.5 appears in the conference paper [43].) The next three sections will show how IS(A) and IS$^\lambda$(A) will offer some different partial answers to Section 1's Paradoxical Question $*$.

4 The IS(A) Axiom System and the Intuition Behind IS$^\lambda$(A)

The next section will prove that the axiom system IS$^\lambda$(A) satisfies a consistency preservation property similar to ISREF(A). Our two goals in this section will be to review the definition of a semantic tableaux proof and to

prove that IS(A) satisfies a consistency preservation property similar to Theorem 3.4's treatment of ISREF(A). Since IS(A) will have a weaker Group-3 schema than $\text{IS}^{\lambda}(A)$, the main theorem in the next section will be strictly stronger than this section's results. The two reasons it is desirable to discuss IS(A) first are that its analysis is substantially simpler than $\text{IS}^{\lambda}(A)$'s analysis, and that all the intermediate results in this section will be necessary interim steps for the next section's main proof.

Our definition of a semantic tableaux proof of a theorem Φ will be similar to that in [10, 23, 32]. Define a **Φ-Based Candidate Tree** for the axiom system α to be a tree structure whose root corresponds to the sentence $\neg\,\Phi$ *rewritten in prenex* normal form* and whose all other nodes are either axioms of α or deductions from higher nodes of the tree. (Unlike Hilbert proofs, the semantic tree notation requires that all axioms be proper sentences, i.e. open formulae are disallowed to appear as axioms in a candidate tree). Let the notation " $\mathcal{A} \implies \mathcal{B}$ " indicate that $\mathcal{B}$ is a valid deduction when $\mathcal{A}$ is an ancestor of $\mathcal{B}$ in the candidate tree T. In this notation, the deduction rules allowed in a candidate tree are:

1. $\Upsilon \wedge \Gamma \implies \Upsilon$ and $\Upsilon \wedge \Gamma \implies \Gamma$.
2. $\neg\neg\Upsilon \implies \Upsilon$. Other valid Tableaux rules for the " $\neg$ " symbol include: $\neg(\Upsilon \vee \Gamma) \implies \neg\Upsilon \wedge \neg\Gamma$, $\neg(\Upsilon \supset \Gamma) \implies \Upsilon \wedge \neg\Gamma$, $\neg(\Upsilon \wedge \Gamma) \implies \neg\Upsilon \vee \neg\Gamma$, $\neg\exists v\, \Upsilon(v) \implies \forall v \neg\Upsilon(v)$, $\neg\forall v\, \Upsilon(v) \implies \exists v\, \neg\Upsilon(v)$ and for any term s the rules $\neg\exists v \leq s\, \Upsilon(v) \implies \forall v \leq s\, \neg\Upsilon(v)$ and $\neg\forall v \leq s\, \Upsilon(v) \implies \exists v \leq s\, \neg\Upsilon(v)$
3. A pair of sibling nodes Υ and Γ is allowed in a candidate tree when their ancestor is $\Upsilon \vee \Gamma$.
4. A pair of sibling nodes $\neg\Upsilon$ and Γ is allowed in a candidate tree when their ancestor is $\Upsilon \supset \Gamma$.
5. $\exists v\, \Upsilon(v) \implies \Upsilon(u)$ where u denotes a newly introduced "Parameter Symbol".
6. $\exists v \leq s\ \Upsilon(v) \implies u \leq s \wedge \Upsilon(u)$ where u denotes a newly introduced "Parameter Symbol" and s denotes a parameter term. (A **Parameter Term** is defined to be any one of a constant symbol, a parameter symbol, or a function symbol whose input arguments can be other parameter terms.)
7. $\forall v\, \Upsilon(v) \implies \Upsilon(t)$ where t denotes a parameter term. The "Parameter Terms" here are built out of any set of constant symbols $c_1, c_2, .., c_m$ and parameter symbols $u_1, u_2, .., u_n$, where each symbol u_i **was previously** introduced by an ancestor of the node storing the new deduction " $\Upsilon(t)$ ".
8. $\forall v \leq s\ \Upsilon(v) \implies t \leq s \supset \Upsilon(t)$ where s and t denote parameter terms (and t satisfies the same constraints as in Item 7).

Define a particular leaf-to-root branch in a candidate tree T to be **Closed** iff it contains both some sentence Υ and its negation $\neg\,\Upsilon$. A **Semantic Tableaux** proof of Φ is defined to be a candidate tree whose root stores the sentence $\neg\Phi$ (written in prenex* normal form) and all of whose root-to-leaf branches are closed. The only distinction between our definition of a semantic tableaux proof and some other conventional definitions in [10, 23, 32, 38] is that we require Φ's proof tree to have its root store $\neg\,\Phi$ *rewritten in prenex**

normal form, whereas some other conventional definitions do not have the prenex* requirement. All our theorems will also hold if we drop the prenex* requirement, but the notation in our main proofs will be simplified if we begin with the assumption that the root has been normalized into prenex* form.

Notation: Let y denote a Semantic Tableaux proof and " $\overline{c_K}$ " designate the constant symbol that formally represents the natural number K . Let us say the symbol $\overline{c_K}$ is **Locally Defined** within the proof y iff at least one proper axiom lying in the proof tree y contains the symbol $\overline{c_K}$. Also, VALUE($\bullet$) will denote a function that maps each term t onto the integer VALUE(t), defined below:

1. VALUE($\overline{c_K}$) $=$ K when $\overline{c_K}$ is "locally defined" in y .
2. VALUE($\overline{c_K}$) $=$ 0 when $\overline{c_K}$ is "locally undefined" in y .
3. VALUE(u) is theoretically allowed to designate any fixed integer when u is a parameter symbol. (However we will often use it in contexts requiring other constraints.)
4. VALUE(t)'s definition will generalize in the natural manner for terms t that contain function symbols F , i.e. VALUE($F(s,t)$) = F(VALUE(s),VALUE(t)).

Define a **Valuation** for the proof-tree y to be a formal function, such as the operation VALUE($\bullet$) above, that maps terms from y's proof-tree onto integer-numbers satisfying the conditions (1)-(4). The symbols ϖ and ϖ^* will often denote particular valuations satisfying these conditions.

Lemma 4.1. Assume that every natural number K, requires at least $\text{Log}_2(K)$ bits to encode the constant symbol " $\overline{c_K}$ " which formally represents it. (We make this assumption because Appendix B's Gödel encoding of IS$^\lambda(A)$ will employ such binary-like representations for encoding constants.). Then a semantic tableaux proof tree y (using either IS(A)'s or IS$^\lambda(A)$'s axioms) will satisfy VALUE($\overline{c_K}$) $<$ $\sqrt{y}$.

Justification of Lemma 4.1. There are several different proofs of Lemma 4.1, depending on what type of Gödel encoding is employed. One possibility is that the encoding of $\overline{c_K}$ will require $2{\cdot}\text{Log}_2(K)$ bits. In this case, Lemma 4.1 is trivially valid because the proof tree y contains only $\text{Log}_2(y)$ bits, implying it certainly has insufficient space to contain a constant symbol representing a number larger than $\sqrt{y}$.

Lemma 4.1 is also valid in the more conservative case where $\overline{c_K}$'s storage requires only $\text{Log}_2(K)$ bits. The proof is then slightly more elaborate. It rests on the fact that even if $\overline{c_K}$'s encoding requires only $\text{Log}_2(K)$ bits, the formal encoding of the Group-Zero, Group-1, Group-2 or Group-3 axiom containing $\overline{c_K}$ will employ at least $2{\cdot}\text{Log}_2(K)$ bits because the relevant axiom will contain other information besides the constant $\overline{c_K}$ (itself). The footnote [5] examines one-by-one all IS(A)'s proper axioms containing the symbol $\overline{c_K}$ and shows

[5] We must separately verify the four cases of Group-Zero, Group-1, Group-2 or Group-3 axioms to verify that each axiom

they all have bit-lengths exceeding $2{\cdot}\text{Log}_2(K)$ This implies that VALUE($\overline{c_K}$) will immediately satisfy Lemma 4.1 when the symbol $\overline{c_K}$ appears inside any of y's proper axioms (by the same trivial argument as had appeared in the preceding paragraph) The second footnote [6] shows that VALUE($\overline{c_K}$) also satisfies Lemma 4.1's requirements when the constant symbol $\overline{c_K}$ never appears within any of y's proper axioms, but it is actually unimportant the reader examine either of these footnotes in detail. This is because all our main theorems will remain valid without any such considerations, if one simply uses the slightly fatter encoding methods from the first paragraph of this proof.

More Notation: Recall that Section 2 indicated that IS(A) will use the axiom given in Equation (17) to recognize that Addition is a total function:

$$\forall x \; \forall y \; \exists z \quad x \; = \; z - y \tag{17}$$

Let t_1 and t_2 denote arbitrary terms built out of Section 2's seven Grounding functions, and let u denote an arbitrary parameter symbol. In our discussion, equations similar to (18), (19) and (20) (below) will be called respectively the **Primary, Secondary and Tertiary deductions** from IS(A)'s addition axiom (17). The reason for this terminology is that by one, two or three applications of the $\forall$-Elimination and $\exists$-Elimination Rules, a Semantic Tableaux proof can derive each of these three equations from (17).

$$\forall y \; \exists z \quad t_1 \; = \; z - y \tag{18}$$

$$\exists z \quad t_1 \; = \; z - t_2 \tag{19}$$

$$t_1 \; = \; u - t_2 \tag{20}$$

Also, let CONS(y) denote the largest quantity VALUE($\overline{c_K}$) among the constant symbols $\overline{c_{K_1}}$, $\overline{c_{K_2}}$ ... $\overline{c_{K_m}}$ appearing in y's proof-tree. (The preceeding sentence referred only to "constant symbols", and CONS(y) thus does not reflect the size of the VALUE(u) quantities for y's parameters and terms.) Let σ denote some

containing a constant $\overline{c_K}$ will have an encoding employing at least $2{\cdot}\text{Log}_2(K)$ bits. Such a case-by-case analysis will reveal that:

1. Each Group-zero axiom will satisfy this property because it will contain a constant symbol $\overline{c_K}$ only if it stores a second constant symbol which requires essentially the same storage space as $\overline{c_K}$.
2. Since the only three constant symbols appearing among the Group-1 axioms are the numbers 0,1 and 2, these axioms obviously always have more than adequately long length.
3. Without loss of generality, we may assume that the symbol $\lceil \Psi \rceil$ in Equation (7) represents the largest constant symbol appearing in this Group-2 axiom. Since Equation (7) contains the formula Φ , in addition to the symbol $\lceil \Phi \rceil$, its formal encoding clearly requires in excess of $2{\cdot}\text{Log}_2(\lceil \Phi \rceil)$ bits.
4. The Group-3 axioms for IS$^{\lambda}$(A) (in Equation (9)) always require in excess of $2\,\text{Log}_2(\lceil \Phi \rceil)$ bits, by the same argument as was given in the case above. (The constants in IS(A)'s Group-3 axiom (from Equation (8)) are also adequately small.)

[6] The reason VALUE($\overline{c_K}$) satisfies Lemma 4.1 even in the extreme case where the constant symbol $\overline{c_K}$ never appears in any of y's proper axioms is simply that VALUE($\overline{c_K}$) will then equal zero, by Part 2 of its definition. This trick obviously shortens the proof of Lemma 4.1, and it may first appear to be cheating by using an overly contrived definition of VALUE($\overline{c_K}$) . However, there is truthfully no harm in using this definition to shorten Lemma 4.1's proof because if y has no proper axiom defining $\overline{c_K}$, then there is certainly no harm in using a notation convention that will then assume VALUE($\overline{c_K}$) = 0.

branch of y's proof tree that consecutively introduces the parameter symbols $u_1,\ u_2,\ u_3,\ ...\ u_n$. Let us say that a valuation ϖ is **Addition-Conservative** over the branch σ iff

1. The parameter u_i will satisfy VALUE(u_i) = VALUE(t_1)+VALUE(t_2) when it is introduced by a tertiary reduction similar to Equation (20).

2. Otherwise, the parameter u_i will always satisfy the inequality

$$\text{VALUE}(u_i) \ \leq \ \text{MAX}[\ \text{CONS}(y),\ \text{VALUE}(u_1), \text{VALUE}(u_2), \text{VALUE}(u_3), ...\text{VALUE}(u_{i-1})] \qquad (21)$$

3. If the sentence " $\exists x\ \Upsilon(x)$ " is valid under the assignment of values indicated by the valuation ϖ and both this sentence and its resulting inference " $\Upsilon(u)$ " appear along the branch σ, then the sentence " $\Upsilon(u)$ " is also valid under ϖ . (The same is also true for the bounded existential sentence " $\exists x \leq s\ \Upsilon(x)$ " and its deduction " $u \leq s\ \wedge\ \Upsilon(u)$ ".)

Let us assume a node N_i on the branch σ has a canonical form similar to either $\phi \vee \psi$ or $\phi \supset \psi$, and a second node N_j on this branch is a deduction from N_i via the $\vee-$elimination or $\supset-$elimination rules. We will say σ has made a *proper sibling choice* relative to the valuation ϖ iff N_j is valid under ϖ *whenever* its ancestor N_i is. Also, the branch σ will be defined to be **Addition-Conservative** iff there exists some Addition-Conservative valuation ϖ for which σ will make all the proper sibling choices.

Lemma 4.2. Suppose y denotes some Semantic Tableaux proof that has an Addition-Conservative branch σ of y. Then each parameter u in this branch will satisfy VALUE(u) $< y$.

Proof Sketch. It is easy to see that the fastest possible growing sequence of parameters $u_0, u_1, u_2, ...u_n$ consecutively introduced along the branch σ will satisfy:

$$\text{VALUE}(u_0) \ = \ \text{CONS}(y) \qquad (22)$$

$$\text{VALUE}(u_{i+1}) \ = \ \text{VALUE}(u_i) + \text{VALUE}(u_i) \qquad (23)$$

For any $i \leq n$, the preceding two equations imply

$$\text{VALUE}(u_i) \ = \ 2^i \cdot \text{CONS}(y) \ \leq \ 2^n \cdot \text{CONS}(y) \qquad (24)$$

We may certainly assume $n < \frac{1}{3}\cdot \text{Log}_2 y$ under Appendix B's method for encoding a proof y (or any other conventional encoding method) because y contains only $\text{Log}_2 y$ bits. Moreover, Lemma 4.1 indicates $\text{CONS}(y) < \sqrt{y}$. Substituting these quantities into (24) yields VALUE(u_i) $< y$. $\square$

The remainder of this section will be devoted to proving the following theorem:

Theorem 4.3 If A is a regularly consistent axiom system then the axiom system IS(A) will be consistent. (In other words, IS($\bullet$) is a Consistency-Preserving Mapping.)

Remark 4.4. This paragraph will sketch the intuition behind Theorem 4.3's proof. (The formal proof of the theorem is a bit long, and it will appear later in this section.) We will begin our more intuitive explanation by considering the Equation (25) below:

$$\forall y \quad \{ \ \alpha(y) \ \supset \ \beta(y) \quad \} \tag{25}$$

Let us say that the Equation (25) is a **Vacuous Truth** iff it satisfies the following two conditions:

a. The Equation (25) is a logically valid statement.

b. Although it is a valid sentence, the Equation (25) will actually not indicate what it may first appear to imply because no element y will actually satisfy either $\alpha(y)$ or $\beta(y)$.

It is well known that "vacuous truths" are often useful intermediate steps appearing in proofs-by-contradiction. (For example, a proof-by-contradiction of the assertion " $\forall y \ \neg\alpha(y)$ can be constructed by simply employing Equation (25)'s "vacuous truth" and showing that no y-element can satisfy $\beta(y)$.) We will now show how one can construct a proof of Theorem 4.3 by using the particular "vacuous truth" given below:

++ Suppose A is regularly consistent. Then there exists a formal mapping method M that has the property that if y is a proof of 0=1 from IS(A) then the mapping's "generated object" $M(y)$ will represent an Addition-Conservative branch in y's proof-tree where there is a parameter u on this branch such that some Gödel number $p \leq$ VALUE(u) will represent **yet another proof** of 0=1 from IS(A).

Both our published 12-page 1993 conference paper [40] and its accompanying more detailed 50-page unpublished manuscript [41] contained intermediate results that were essentially equivalent to the Assertion ++. It also should be noted that the Lemma 4.8 (appearing later in this section) will be essentially a result that is strictly stronger than the Assertion ++ . At the present juncture, we shall ask the reader to put aside such considerations, and to just temporarily assume the validity of the Assertion ++ . The remainder of this paragraph will explain how the Assertion ++ can enable us to formulate a very simple and direct proof of Theorem 4.3. It will rest on the following two quite simple observations:

1. The *combination* of Assertion ++ and Lemma 4.2 allows one to rewrite the "vacuous truth" from Assertion ++ into a second slightly reworded and *more succinct* form. This rephrased vacuous truth shall state that *"any proof y of 0=1 from* IS(A) *will be automatically associated with an ordered pair* (p, u) *where* $p \leq$VALUE(u) $< y$ *and p represents ANOTHER proof of 0=1".*

2. It is possible to prove Theorem 4.3 by using a method of proof-by-contradiction. It will begin by hypothesizing that IS(A) is inconsistent. It will thus let y represent the least integer that is a proof of 0=1 from IS(A). From Item 1 (above), the preceding is impossible because Item 1 indicates that some $p < y$ must then represent another proof of 0=1. This contradicts y's assumed minimality, and enables our proof-by-contradiction to immediately reach its desired end.

We wish to close this paragraph by emphasizing that the items (1) and (2) above are basically trivial arguments. Thus, they show that after one has established the validity of the somewhat complicated "vacuous truth" given in Assertion ++, the remainder of the proof of Theorem 4.3 is fairly routine and easy.

Remark 4.5. Before turning to a more detailed discussion of Theorem 4.3's properties, it is desirable to explain why our axiom system IS(A) had omited the assumption that Multiplication is a total function. The inclusion of such a Multiplication axiom would have caused the proof sketched in the preceding paragraph to break down completely. The reason for this collapse is easiest to explain if we employ the following notation:

a. ISMULT(A) will denote the analog of the axiom system IS(A) that recognizes Multiplication as a total function. Its Group-3 axiom will thus assert that "*No Semantic Tableaux proof of 0=1 exists from* ISMULT(A)".

b. The definition of a Multiplication Conservative valuation ϖ over a branch σ will be similar to an Addition Conservative valuation except that Part-1 of the definition will now also allow a parameter u_i to possibly satisfy the multiplicative identity: VALUE(u_i) = VALUE(t_1) $*$ VALUE(t_2).

At first it might appear that the analog of Remark 4.4's 2-part proof of Theorem 4.3 would also apply to ISMULT(A) because the latter does in fact satisfy an analog of statement ++ (with the phrase "Addition Conservative" valuation simply replaced by the phase "Multiplication Conservative"). However, Remark 4.4's proof of Theorem 4.3 does indeed break down fully for ISMULT(A). The difficulty arises because ISMULT(A) contains no analog for Lemma 4.2 (in step 1 of Remark 4.4's 2-part proof). Lemma 4.2 essentially indicated that every parameter u lying on an Addition-Conservative Branch automatically satisfied VALUE(u) $< y$ under the IS(A) axiom system. The analog of this inequality for ISMULT(A) is simply invalid for Multiplication-Conservative branches because multiplication allows a sequence of parameters $u_0, u_1, u_2, ...u_n$ to grow at a *very much, much faster rate.* (See footnote[7] for an example.) This is the intuitive reason that Theorem 4.3 is valid for IS(A), but the analogs of its proof for ISMULT(A) are simply not available.

One formal proof of the consistency of IS(A), using approximately the methods sketched in Remark 4.4, was given in the technical report [41] – with a summary of its proof appearing in the more abbreviated

[7]The sequence $u_0, u_1, u_2, ...u_n$ can permit VALUE(u_n) $> y$ in a proof tree y if $u_0 = 2$, $u_{i+1} = {u_i}^2$ and then VALUE(u_n) $= 2^{2^n}$

12-page conference paper [40]. If the only goal of this paper was to prove IS(A)'s consistency, we would use "Addition-Conservative" branches to directly prove the validity of Assertion ++ . However, our goals are much broader because we need an umbrella formalism that will also provide a short proof of the consistency of the more general IS$^{\lambda}(A)$ system (of Section 5). Therefore, the version of proof for Theorem 4.3, given below, will be more complicated than the idea sketched in Remark 4.4, but its machinery will at the same time be much more powerful. In particular, the machinery in Theorem 4.3's proof will employ the notion of a "$(k,m)-$Affirmative Branch" (defined in the next paragraph) rather than an "Addition-Conservative" branch (from Remark 4.4) because we will need the added power of $(k,m)-$Affirmativeness to prove the generalizations of Theorem 4.3 that will appear in Section 5.

Definitions: Recall that branches in Candidate-Trees (unlike Semantic Tableaux Proofs) are not required to be "closed" (by containing a pair of contradictory sentences). Let σ denote a branch in a candidate tree and ϖ denote some fixed valuation function. Given a sentence Ψ on this branch σ , let $\overbrace{\Psi}$ denote a sentence identical to Ψ except that each parameter symbol u in Ψ is replaced by the constant VALUE(u). Also, $\overbrace{\Psi \Downarrow m}$ will denote a sentence identical to $\overbrace{\Psi}$ except that its unbounded quantifiers are changed to quantifiers bounded by m . (Bounded quantifiers do not have their range change). Define a branch σ in a candidate-tree to be **(k,m)-AFFIRMATIVE** iff there exists some valuation ϖ such that:

I. Unless a sentence Ψ lying on the branch σ is a "**Special Exception**", the sentence $\overbrace{\Psi \Downarrow m}$ is required to be valid. Here the "Special Exceptions" are defined to be the axiom in Equation (17) (which had specified Addition was a a total function) and also its Primary and Secondary deductions, given in Equations (18) and (19). (Tertiary Deductions will *not* be "Special Exceptions".)

II. Let β_i denote the number of bits used by Appendix B to formally encode the sentences lying in the first i depth levels in the branch σ , and N_i denote the particular node on the branch σ at depth i. Each parameter symbol u appearing in the node N_i of the proof y will satisfy

$$\text{VALUE}(u) \; < \; (\, k \, + \, \sqrt{y}\,) \cdot 2^{[\,\frac{1}{6}\beta_i\,]\,-\,2}. \tag{26}$$

Lemma 4.6 Suppose y is a candidate-tree containing a (k,m)-Affirmative Branch σ . Then y cannot be a semantic tableaux proof using either the axiom system IS$^{\lambda}(A)$ or IS(A).

Proof. Suppose for the sake of contradiction that y is a semantic tableaux proof. Then by definition, every branch of y , including σ, will contain some sentence Ψ and its negation $\neg\,\Psi$. The footnote [8]

[8]Recall that we define a sentence to be Π_1^- if it is of the approximate form $\forall v_1\,\forall v_2\,...\forall v_j\;\varphi(v_1,v_2,...v_j)$ where $\varphi(v_1,v_2,...v_j)$ is a Δ_0^- formula. Note that all of both IS(A)'s and IS$^{\lambda}(A)$'s proper axioms are such Π_1^- sentences, except for the Equation (17) (which was Π_2^-). Moreover, the root of a Φ-based candidate tree consists of a sentence equivalent to $\neg\Phi$, *rewritten in*

proves that Ψ must then be a Δ_0^- sentence. This is important because no Δ_0^- sentence is one of Item I's allowed "Special Exceptions". Thus since both Ψ and its negation $\neg\,\Psi$ appear along an affirmative branch σ, Item I automatically requires both $\overbrace{\Psi \Downarrow m}$ and $\overbrace{\neg\Psi \Downarrow m}$ to be true.

However, it is impossible that both $\overbrace{\Psi \Downarrow m}$ and $\overbrace{\neg\Psi \Downarrow m}$ are simultaneously true. Therefore our proof-by-contradiction has reached its desired end. It has shown that the assumption that y was a semantic tableaux proof, *containing a* (k,m)*-Affirmative branch,* led to a contradiction. □

Lemma 4.7. Let σ_d denote the fragment of a branch σ whose nodes have depth $\leq d$. Assume each node N_j in this fragment satisfies Part-ii of the definition of an Affirmative Branch. Then any term t in the proof y whose parameters are introduced on the sub-branch σ_d satisfies VALUE$(t) < (k+\sqrt{y}) \cdot 2^{[\frac{1}{6}\beta_d]-2}$.

Proof. The justification of Lemma 4.7 is extremely trivial and straightforward. However, the formal proof of Lemma 4.7 should be given because Lemma 4.7 is used on five occasions during the next two sections. Lemma 4.7's proof begins with the following two simple observations:

1. Lemma 4.1 indicates that any constant c appearing in the proof y will satisfy VALUE$(c) < \sqrt{y}$.
2. Part-ii of the definition of a (k,m)-Affirmative Branch implies that all the parameters u lying on the sub-branch σ_d satisfy VALUE$(u) < (k+\sqrt{y}) \cdot 2^{[\frac{1}{6}\beta_d]-2}$.

We may assume $\beta_d \geq 12$ because a single sentence will clearly require more than twelve bits for encoding it, under all conventional encoding methods, including the particular encoding method employed in Appendix B. Since our seven Grounding functions are non-growth functions, any term built out of the constants and parameters (above) will obviously also respect the bound: VALUE$(t) < (k+\sqrt{y}) \cdot 2^{[\frac{1}{6}\beta_d]-2}$. □

Our next lemma will have a somewhat unusual quality. Its opening sentence will state "Let us temporarily assume that Theorem 4.3 is false and ...". The perspective of Lemma 4.8 is useful because the proof of Theorem 4.3 will be a proof by contradiction. The "temporary assumption" that Theorem 4.3 is false will help establish the needed contradiction in Theorem 4.3's proof.

Lemma 4.8. Let us temporarily assume that Theorem 4.3 is false. Therefore, let A be a regularly consistent axiom system such that IS(A) is inconsistent. Let y denote the smallest integer which encodes a semantic tableaux proof tree of 0=1 from IS(A). Also, define $m = y-1$ and $k = 0$. Then the tree y must contain a (k,m)-Affirmative Branch.

Most of the remainder of this section will be devoted to proving Lemma 4.8. However before proving

prenex normal form.* All these founding sentences thus have their " ¬ " symbols pushed to the right of all their unbounded quantifiers. This makes it very difficult for a deduction from these sentences, " $\neg\,\Psi$ " to have the " ¬ " symbol located in its leftmost position. In particular, the semantic tableaux deduction rules (1-6) will allow a sentence " ¬Ψ " to appear in the string of deductions arising from such tightly normalized founding sentences only when Ψ is Δ_0^-.

Lemma 4.8, we wish to show how it will provide us with a pleasantly short 5-sentence proof of Theorem 4.3:

Proof of Theorem 4.3. Suppose for the sake of contradiction that Theorem 4.3 is false (i.e. there exists a Regularly Consistent axiom system A such that IS(A) is inconsistent). Let y denote the smallest integer which encodes a semantic tableaux proof tree of 0=1 from IS(A). By Lemma 4.8, the tree y "*must contain*" a (k, m)-Affirmative Branch. However, Lemma 4.6 prohibits a proof tree from containing any (k, m)-Affirmative Branch (because it indicates the presence of an Affirmative Branch will preclude a candidate tree from representing a proof). Since it is impossible for both these contradictory conditions to be simultaneously valid, our proof-by-contradiction has shown it is impossible for Theorem 4.3 to be false. □

Our final goal is to prove Lemma 4.8. Let N_i denote the node in the branch σ whose depth = i . Let σ_i denote the fragment of the branch σ that lies between its root node N_0 and the node N_i . We will say that the fragment σ_i satisfies the *(k, m)-Affirmative Condition* iff every node $N_0, N_1, N_2, ..., N_i$ on this fragment satisfies the two requirements (I) and (II) mentioned earlier. The proof of Lemma 4.8 will establish the existence of an (k, m)-Affirmative Branch σ by describing a process that inductively verifies that each fragmented branch σ_i in the sequence $\sigma_0, \sigma_1, \sigma_2, \sigma_3, ...$ is (k, m)-Affirmative.

Before starting the proof of Lemma 4.8, the function of Part-II of the definition of an $(k, m)-$Affirmative Branch should be briefly explained. This part of the definition of an affirmative branch *was never used in* in the proof of Lemma 4.6 or in the proof of Theorem 4.3. The reason we need the Part-II Condition is that the proof below becomes simpler when it shows that a branch σ satisfies *both* the Parts I and II conditions (together), than would be the case if it established just Part-I alone. (This is because the strengthened inductive hypothesis will significantly shorten the eighth step in Lemma 4.8's inductive proof).

Proof of Lemma 4.8. Our inductive proof will be divided into eleven cases. In each case, the (k, m)-Affirmative Branch is assumed to employ parameters $k = 0$ and $m = y - 1$. The reason that the proof (below) refers to a "(k, m)" rather than "$(0, y - 1)$" constrained branch is that the next section's proof of the stronger Theorem 5.1 will employ different values for (k, m) . The proof of Lemma 4.8 will thus be more portable if the notation "(k, m)" rather than "$(0, y - 1)$" is employed.

Assuming the node N_{i-1} (at the bottom of σ_{i-1}) is not a leaf, each of the eleven cases (below) will show there exists a longer (k, m)-Affirmative fragment σ_i, which contains some additional node N_i :

1. The Case where N_i designates the root node of Lemma 4.8's tree y: Since y represents a proof of 0=1, the root of the semantic tableaux tree y is defined to be the sentence " $0 \neq 1$ ". Since " $0 \neq 1$ " is a valid sentence containing no quantifiers, $\overbrace{\{0 \neq 1\} \Downarrow m}$ is obviously also valid. Hence, this sentence automatically satisfies Part-I of the definition of a (k, m)-Affirmative Branch. Since no parameter

symbols u appear in this sentence, it satisfies Part-II of this definition, trivially, by default. $\square$

2. The Case where N_i represents a Group-zero, Group-1 or Group-2 axiom of IS(A): We need not consider the special case where N_i stores a sentence Ψ, corresponding to Equation (17)'s Group-1 axiom (which had indicated Addition was a total functions). This is because Part-I of the definition of (k , m)-Affirmative Branch indicates this sentence is a "Special Exception", where $\overbrace{\Psi \Downarrow m}$ is not required to be satisfied. Leaving aside this "Special Exception", all the other Group-zero, Group-1 and Group-2 axiom are Π_1^- sentences. Clearly, every Group-zero and Group-1 axiom is valid in the Standard Model. Also, every Group-2 axiom is valid in the Standard Model of the Natural Numbers (by our assumption that A is regularly consistent). Thus, these facts imply that $\overbrace{\Psi \Downarrow m}$ is valid (simply because Ψ is a valid Π_1^- sentence). Hence, every Group-zero, Group-1 or Group-2 axiom meets Part-I of the definition of an Affirmative Branch.

Since there are no parameter symbols u appearing in proper axioms, Part-II of the definition of an Affirmative Branch is again trivially satisfied, by default. $\square$

3. The Case where N_i represents the Group-3 axiom of IS(A): Let Γ denote this Group-3 axiom. It was formally defined in Section 2 to be the sentence:

$$\forall\, p \quad \neg\, \mathrm{SemPrf}_{\mathrm{IS(A)}} \left(\ulcorner\, 0 = 1 \,\urcorner\ ,\ p \right) \tag{27}$$

This case differs from Case 2 because the hypothesis of Lemma 4.8 does not imply Γ is a valid sentence. However, Lemma 4.8's hypothesis does imply $m+1$ is the smallest proof of 0=1 from IS(A). Thus even if Γ is invalid, the sentence $\overbrace{\Gamma \Downarrow m}$ is valid. Hence, Part-I of the definition of a $(k, m)-$Affirmative Branch is automatically satisfied. Since no parameter symbols u appear in the sentence Γ , Part-II of this definition is once again trivially satisfied by default. (It is extremely important the reader examine footnote [9].) $\square$

4. The Proof for the Case where N_i is generated by the $\wedge$-Elimination Rule (defined in the second paragraph of this section): Trivial because a sentence Υ (stored in the node N_i) will automatically satisfy the (k, m)-Affirmative Condition when some ancestor of it storing the sentence $\Upsilon \wedge \Theta$ does. $\square$

5. The Proof for the Case where N_i is generated by a $\neg$-Deduction Rule: Once again, it is trivial that a sentence Υ (stored in the node N_i) will satisfy the (k, m)-Affirmative Condition when some ancestor of it storing the sentence $\neg\neg\Upsilon$ does. An identical argument also applies to the other seven variants of $\neg$-Deduction Rules (described in item 2 of the second paragraph of this section). $\square$

[9] The proof of Case 3 was only five sentences long and very straightforward. Yet this simplicity can be very misleading because Case 3 is perhaps the central case. The subtle aspect of Case 3 is that it will be meaningful only if the other cases assure that the valuation ϖ is unable to represent any term t along the branch σ where $\mathrm{VALUE}(t) > m$. (Otherwise, Case 3's sentence " $\overbrace{\Gamma \Downarrow m}$ " would be simply useless.) As the reader examines the other cases, he should remember each case implicitly assures $\mathrm{VALUE}(t) \leq m$ when it establishes its node N_j satisfies the $(k, m)-$Affirmative Condition. (This is intuitively because $k = 0$, $m = y - 1$ and $\beta_j \leq \mathrm{Log}_2(y)$, together with Lemma 4.7, implicitly assure $\mathrm{VALUE}(t) \leq m$ holds for every term t .)

6. The Proof for the Case where N_i is generated by the $\vee$-Elimination Rule: A $\vee$-Deduction consists of introducing a pair of sibling nodes Υ and Θ when these nodes have a common ancestor $\Upsilon \vee \Theta$. Since the inductive hypothesis implies $\Upsilon \vee \Theta$ satisfies the (k, m)-Affirmative Condition, one of Υ or Θ must also satisfy this condition. Our algorithm for constructing the branch σ_i will have N_i designate that particular one of the two nodes Υ or Θ which satisfies the (k, m)-Affirmative Condition. $\square$.

7. The Proof for the Case where N_i is generated by the $\supset$-Elimination Rule: Essentially the same as the preceding paragraph. $\square$

8. The Proof for the Case where N_i is generated by the (Unbounded version of the) $\forall$-Elimination Rule: Assuming some ancestor N_a of N_i stores the sentence $\forall v\, \Upsilon(v)$, the $\forall$-Elimination Rule allows N_i to indicate the sentence $\Upsilon(t)$ (where t denotes some parameter term).

The combination of the inductive hypothesis and Lemma 4.7 trivially implies:

$$\text{VALUE}(t) \ < \ (k + \sqrt{y}) \cdot 2^{[\,\frac{1}{6}\beta_{i-1}\,]-2} \tag{28}$$

Since β_{i-1} represents the bit-length of the part of proof y that lies along the path σ_{i-1}, we have:

$$\beta_{i-1} \ < \ \text{Log}_2(y) \tag{29}$$

Since Lemma 4.8's hypothesis indicates $m = y - 1$ and $k = 0$, Equations (28) and (29) trivially imply

$$\text{VALUE}(t) \ \leq \ m \tag{30}$$

The remainder of our proof of the Case 8 will be divided into the two sub-cases where:

A. The ancestor N_a of N_i which stores "$\forall v\, \Upsilon(v)$" **IS** one of the Affirmative Branch's "Special Exceptions"

B. This ancestor N_a **IS NOT** one of the Affirmative Branch's "Special Exceptions"

In both cases, we will show that N_i satisfies the $(k, m)-$Affirmative Condition.

Justification for the Sub-Case (A): The "Special Exceptions" were carefully defined so that $\Upsilon(t)$ will automatically be a Special Exception when $\forall v \Upsilon(v)$ is a Special Exception. Hence, it is unnecessary to show that the Sub-Case (A) satisfies Part-I of the definition of an (k, m)-Affirmative Branch because the node N_i is another permitted "Special Exception", where the Part-I requirement is not required. Our proof thus needs only to show N_i satisfies Part-II of the definition of an (k, m)-Affirmative Branch

The proof justifying the Part-II requirement is basically trivial. This is because the $\forall$-Elimination Rule requires each parameter u in the node N_i's sentence $\Upsilon(t)$ to appear "previously" in a higher position on

the sub-branch σ_{i-1}. It can thus be inductively assumed that $\text{VALUE}(u) < (k+\sqrt{y}) \cdot 2^{[\frac{1}{6}\beta_{i-1}]-2}$. Since $\beta_{i-1} < \beta_i$, this inequality immediately implies:

$$\text{VALUE}(u) < (k+\sqrt{y}) \cdot 2^{[\frac{1}{6}\beta_{i-1}]-2} < (k+\sqrt{y}) \cdot 2^{[\frac{1}{6}\beta_i]-2} \tag{31}$$

The latter shows that u satisfies the Part-II condition.

Justification for the Sub-Case (B): Since the ancestor N_a of N_i is not a "Special Exception" in this Sub-Case, we can assume that the sentence $\overbrace{\forall v \Upsilon(v) \Downarrow m}$ is valid (because otherwise N_a would violate the inductive hypothesis). The combination of the preceding overbrace expression and Equation (30) imply $\overbrace{\Upsilon(t) \Downarrow m}$ is also valid. Thus, the Part-I requirement is satisfied. The justification of the Part-II requirement in the Sub-Case (B) is identical to its justification for the Sub-Case (A) (in the paragraph above). □

9. The Proof for the Case where N_i is generated by the (Bounded version of the) $\forall$-Elimination Rule: The justification of the Part-I requirement is essentially identical to Sub-Case B of Case 8. Thus in this case, an ancestor N_a of N_i stores a sentence of the form "$\forall v \leq s\, \Upsilon(v)$". The inductive hypothesis allows us to presume that $\overbrace{\forall v \leq s\ \Upsilon(v) \Downarrow m}$ is valid. As before, the preceding overbrace expression together with Equation (30) imply $\overbrace{t \leq s \supset \Upsilon(t) \Downarrow m}$ (and hence that Part-I of the definition of an Affirmative Branch is satisfied). Also it is *again trivial* to show that N_i satisfies the Part-II requirement (by once again using the construction from the second paragraph in Sub-Case (A) of Case 8). □

10. The Proof for the Case where N_i is generated by the (Bounded version of the) $\exists$-Elimination Rule: Assuming some ancestor N_a of N_i stores the sentence $\exists v \leq s\ \Upsilon(v)$, the rule for $\exists$-Elimination allows N_i to indicate the sentence $u \leq s \wedge \Upsilon(u)$ (where u denotes a new unused parameter symbol). Applying Lemma 4.7 and the inductive hypothesis to N_a implies that there exists a (k,m)-Affirmative Valuation ϖ over the sub-branch σ_{i-1} where both $\overbrace{\exists v < s\ \Upsilon(v) \Downarrow m}$ and the inequality $\text{VALUE}(s) < (k+\sqrt{y}) \cdot 2^{[\frac{1}{6}\beta_{i-1}]-2}$ are valid.

We need to construct an extended valuation ϖ^* (over the sub-branch σ_i) which differs from ϖ only by defining a new quantity VALUE(u) for N_i's new parameter u . The last sentence in the preceding paragraph implies that such a ϖ^* can be constructed where both $\overbrace{\Upsilon(u) \Downarrow m}$ and $\text{VALUE}(u) < (k+\sqrt{y}) \cdot 2^{\frac{1}{6}\beta_i - 2}$ are valid. Hence, the node N_i will satisfy both Parts I and II of the $(k,m)-$Affirmative Condition. □

11. The Proof for the Case where N_i is generated by the (Unbounded version of the) $\exists$-Elimination Rule: Assuming some ancestor N_a of N_i stores the sentence $\exists v\, \Upsilon(v)$, the $\exists$-Elimination Rule allows N_i to indicate the sentence $\Upsilon(u)$ (where u denotes a new unused parameter symbol). This case is quite simple because the only axiom of IS(A) which contains an *unbounded existential quantifier* is

the Equation (32) (below), which indicates that Addition is a total function. (Any other type of existential quantifier that appears in a proof of $0 = 1$ from IS(A) will be *bounded !!!*)

$$\forall x \; \forall y \; \exists z \quad x \; = \; z - y \tag{32}$$

Below in (33) and (34) are the "Secondary" and "Tertiary" deductions generated from Equation (32). Note (33) will lie on the sub-branch σ_{i-1} whenever (34) represents the sentences stored in N_i . Moreover, the combination of (33)'s presence on the sub-branch σ_{i-1} , the inductive hypothesis and Lemma 4.7 together imply the validity of Equations (35) and (36).

$$\exists z \quad t_1 \; = \; z - t_2 \tag{33}$$

$$t_1 \; = \; u - t_2 \tag{34}$$

$$\text{VALUE}(t_1) \; < \; (\, k \, + \, \sqrt{y}\,) \cdot 2^{\,[\,\frac{1}{6}\beta_{i-1}\,]\,-\,2} \tag{35}$$

$$\text{VALUE}(t_2) \; < \; (\, k \, + \, \sqrt{y}\,) \cdot 2^{\,[\,\frac{1}{6}\beta_{i-1}\,]\,-\,2} \tag{36}$$

Let us consider a valuation ϖ satisfying:

$$\text{VALUE}(u) \; = \; \text{VALUE}(t_1) + \text{VALUE}(t_2) \tag{37}$$

The combination of Equations (35), (36) and (37) implies:

$$\text{VALUE}(u) \; < \; 2 \; \cdot \; (\, k \, + \, \sqrt{y}\,) \cdot 2^{\,[\,\frac{1}{6}\beta_{i-1}\,]\,-\,2} \tag{38}$$

The formal Gödel encoding of the sentence (34) obviously requires substantially more than six bits for encoding it, under all conventional encoding methods, including the particular encoding method employed in Appendix B. In our notation, this implies that $\beta_i \, - \, \beta_{i-1} \; \geq \; 6$. Substituting the latter inequality into (38), we get:

$$\text{VALUE}(u) \; < \; (\, k \, + \, \sqrt{y}\,) \cdot 2^{\,[\,\frac{1}{6}\beta_{i}\,]\,-\,2} \tag{39}$$

This implies that the node storing the sentence (34) will satisfy the $(k, m)-$Affirmative Condition (because Equation (37) shows the Part-I condition is satisfied and Equation (39) shows Part-II is satisfied). $\square$

5 The Consistency of IS$^{\lambda}(A)$

This section will prove IS$^{\lambda}(A)$'s consistency property. The proof of Theorem 5.1 will be pleasantly short because it will be based on incrementally revising Section 4's proof of Theorem 4.3.

Theorem 5.1 This theorem will consider the version of IS$^{\lambda}(A)$ that employs a constant $\lambda = \frac{3}{4}$ Suppose that A is a regularly consistent axiom system. Then IS$^{\lambda}(A)$ is also consistent.

Comment: We have convinced ourselves Theorem 5.1 is also valid for any $\lambda > .01$, and probably for much smaller λ. The reason Theorem 5.1 assumed $\lambda = \frac{3}{4}$ was simply to make its proof shorter.

Our proof of Theorem 5.1 will begin with two preliminary lemmas. These lemmas will have an unusual quality because Theorem 5.1 will employ a proof by contradiction. Thus, one of Lemma 5.2's opening sentences will state "Suppose Theorem 5.1 is false...". Lemma 5.3's statement will similarly ask the reader to "temporarily assume" a condition that will be later shown to be impossible. The perspectives of Lemmas 5.2 and 5.3 will be helpful for proving Theorem 5.1 because the final proof will be a proof by contradiction.

Lemma 5.2. Consider the version of the $\mathrm{IS}^{\lambda}(A)$ axiom system that uses a constant $\lambda = \frac{3}{4}$. Suppose that Theorem 5.1 is false for this version of the $\mathrm{IS}^{\lambda}(A)$ axiom system. (In other words, suppose that there exists an axiom system A such that A is regularly consistent but $\mathrm{IS}^{\lambda}(A)$ is inconsistent.) Then there will exist a tuple (x, y, z, Φ) where Φ is a prenex* sentence satisfying:

$$\mathrm{SemPrf}_{\mathrm{IS}^{\lambda}(A)} (\ulcorner \Phi \urcorner , y) \quad \wedge \quad y^{\frac{3}{4}} < \frac{z}{x} \quad \wedge \quad \neg (\Phi_z^x) \tag{40}$$

Proof. Since $\mathrm{IS}^{\lambda}(A)$ is inconsistent, at least one of the axioms of $\mathrm{IS}^{\lambda}(A)$ must be invalid under the Standard Model of the Natural Numbers. If A is regularly consistent then all $\mathrm{IS}^{\lambda}(A)$'s Group-2 axioms must be valid in the Standard Model. Also, its Group-zero and Group-1 axioms are clearly valid. Therefore the inconsistency of $\mathrm{IS}^{\lambda}(A)$ will imply that at least one of its Group-3 axioms is invalid in the Standard Model. Examining the generic form of $\mathrm{IS}^{\lambda}(A)$'s Group-3 axiom (described in Equation (9)) and using our assumption that some Group-3 axiom (with $\lambda = \frac{3}{4}$) is false, we are forced to conclude that there must exist a tuple (x, y, z, Φ) where Φ is a prenex* sentence satisfying (40). □

Lemma 5.3. Suppose Φ is a prenex* sentence, the tuple (x, y, z, Φ) satisfies Equation (40), and A is a regularly consistent axiom system. Also, suppose that every other tuple (x^*, y^*, z^*, Φ^*) satisfying (40) has its third component z^* satisfying $z^* \geq z$. (It will be proven later that no such tuple (x, y, z, Φ) will satisfy these requirements, but let us temporarily assume that some such (x, y, z, Φ) does exist.) These conditions will then imply that the integer y , viewed as a Gödel number describing a tree, will contain at least one (k, m)-Affirmative Branch (with constants $k = x$ and $m = z - 1$).

The proof of Lemma 5.3 is analogous to Lemma 4.8's proof. It appears at the end of this section. Our immediate goal is to explain how Lemmas 4.6, 5.2 and 5.3 will provide a short 10-sentence proof of Theorem 5.1. It is given below:

Proof of Theorem 5.1. Suppose for the sake of contradiction that Theorem 5.1 was false for the version of the $\mathrm{IS}^{\lambda}(A)$ formalism that sets the constant $\lambda = \frac{3}{4}$. Then by Lemma 5.2, there must exist some tuple

(x, y, z, Φ) that will satisfy (40).

Hence trivially, there must then exist some such tuple with *minimal* value in its third component. By this, we simply mean that there will exist a tuple (x, y, z, Φ) which will satisfy (40) and has the property that no other tuple (x^*, y^*, z^*, Φ^*) can also satisfy (40) unless $z^* \geq z$.

Such a tuple will satisfy the hypothesis of Lemma 5.3. The combination of Lemmas 4.6 and 5.3 then easily implies (see footnote[10]) y *cannot* represent the Gödel number of a Semantic Tableaux proof.

However, Lemma 5.2's Equation (40) blatantly contradicts the preceding sentence. (This is because its formula " $\text{SemPrf}_{\text{IS}^{\lambda}(A)}$ ($\ulcorner \Phi \urcorner$, y) " indicates that y is a semantic tableaux proof.) This contradiction completes our proof-by-contradiction. In particular, it shows that Theorem 5.1 must be true because otherwise two inherently incompatible conditions would be simultaneously valid. □

We will now complete the proof of Theorem 5.1 by showing that Lemma 5.3 is valid:

Proof of Lemma 5.3. The existence of a Affirmative Branch σ will be established using an inductive argument, analogous to Lemma 4.8's proof. Assuming that a node N_{i-1} at the bottom of a (k, m)-Affirmative sub-branch σ_{i-1} is not a leaf, we will show how it is possible to construct a longer sub-branch σ_i , containing the additional node N_i, which is also (k, m)-Affirmative. One difference between the proofs of Lemmas 4.8 and 5.3 is that the former used constants k=0 and m=$y - 1$, whereas we will now use constants k=x and m=$z - 1$. (However, this distinction is quite minor.)

The proof of Lemma 5.3 is divided into eleven cases, similar to Lemma 4.8's proof. Six of these eleven cases will be verbatim identical to Lemma 4.8's treatment. Three more will differ only slightly from Lemma 4.8's proof analysis. Therefore our discussion of the eleven cases will delineate only those aspects of Lemma 5.3's proof that use a different justification than Lemma 4.8's analog. In the inductive proof (below), we assume that the sub-branch $\sigma_{i-1} = \{N_0, N_1, N_2, ...N_{i-1}\}$ is (k, m)-Affirmative: Assuming N_{i-1} is not a leaf, our proof (below) shows how it is possible to construct a longer sub-branch σ_i , containing the additional node N_i , which has a valuation that is (k, m)- Affirmative:

1. The Proof for the Case where N_i designates the root node of a Semantic Tableaux Proof Tree: Let the generic notation " [Υ] " denote Υ written in prenex* form. Thus the root of our Φ-based candidate tree will store the sentence [$\neg\Phi$] (according to Section 4's definition).

We will first show that this sentence satisfies Part-I of the (k, m)-Affirmative Condition. Since Φ_z^x is

[10]The implication holds because Lemma 5.3 indicates that the tree encoded by the integer y contains at least one Affirmative Branch. In this context, Lemma 4.6 disallows y from representing a Semantic Tableaux proof.

assumed to be prenex* , it is permissible to employ the identity:

$$\neg\,(\Phi_z^x) \quad \equiv \quad [\ \neg\Phi\ ]_x^z \tag{41}$$

It is easy to verify that $x \leq z-1$ (since (40) indicated that $y^{\frac{3}{4}} < z/x$ and since every integer y, which represents the Gödel number of a proof, is certainly greater than say 4, under any reasonable encoding scheme). The $x \leq z-1$ inequality, combined with Lemma 3.1A, immediately implies

$$[\ \neg\Phi\ ]_x^z \quad \supset \quad [\ \neg\Phi\ ]_{z-1}^z \tag{42}$$

Also, it follows directly from Lemma 3.1B that

$$[\ \neg\Phi\ ]_{z-1}^z \quad \supset \quad [\ \neg\Phi]_{z-1}^{z-1} \tag{43}$$

Since the root node's sentence $[\ \neg\Phi\ ]$ contains no parameter symbol u , our notation convention implies:

$$[\ \neg\Phi]_{z-1}^{z-1} \quad \equiv \quad \overbrace{[\ \neg\Phi\ ] \Downarrow [z-1]} \tag{44}$$

The hypothesis of Lemma 5.3 indicated that $\neg\,(\Phi_z^x)$ was valid (ergo $\neg\,(\Phi_z^x)$ was one of the conditions indicated by Equation (40)). Since the hypothesis of Lemma 5.3 indicates $m = z-1$, the combination of $\neg\,(\Phi_z^x)$ with the four identities (above) immediately implies that $\overbrace{[\ \neg\Phi\ ] \Downarrow m}$ is valid.

Hence, the sentence stored in the root satisfies Part-I of the requirement for σ to be an (k,m)-Affirmative Branch. Since there are no parameter symbols u appearing in this sentence, it must also satisfy Part-II of the definition of (k,m)-Affirmative Branches, simply by default. □

2. The Proof for the Case where N_i represents a Group-zero, Group-1 or Group-2 axiom of $\mathbf{IS}^{\lambda}(A)$: Verbatim Identical to Case 2 in Lemma 4.8's proof. □

3. The Proof for the Case where N_i represents a Group-3 axiom of $\mathbf{IS}^{\lambda}(A)$: Let Γ_{Υ} denote the Group-3 axiom for the sentence Υ and the fixed constant $\lambda = \frac{3}{4}$. It is formally defined below:

$$\forall\, x\ \forall\, y\ \forall\, z \quad \{ \quad \text{SemPrf}_{\text{IS}^{\lambda}(A)}\ (\ \ulcorner\, \Upsilon\, \urcorner\ ,\ y\ ) \quad \wedge \quad y^{3/4} < \frac{z}{x} \quad \supset \quad \Upsilon_z^x \quad \} \tag{45}$$

If Equation (45) is invalid, there must exist a tuple (a,b,c,Υ) satisfying

$$\text{SemPrf}_{\text{IS}^{\lambda}(A)}\ (\ \ulcorner\, \Upsilon\, \urcorner\ ,\ b\ ) \quad \wedge \quad b^{\frac{3}{4}} < \frac{c}{a} \quad \wedge \quad \neg\,(\Upsilon_c^a) \tag{46}$$

The minimality assumption (from the hypothesis of Lemma 5.3) then implies that $c \geq z$ in Equation (46). This inequality assures the sentence $\overbrace{\Gamma_{\Upsilon} \Downarrow [z-1]}$ is valid, *even if* Γ_{Υ} is invalid! Since the hypothesis of Lemma 5.3 indicates that $m = z-1$, Part-I of the definition of an (k,m)-Affirmative Branch is thus satisfied. Since there are no parameter symbols u appearing in the sentence Γ_{Υ} , Part-II of the definition of an (k,m)-Affirmative Branches is trivially satisfied, again by default. □

4, 5, 6 and 7. The Proof for the Cases where N_i is generated by one of an $\wedge$-Elimination, $\neg$-Elimination, $\vee$-Elimination, or $\supset$-Elimination, Rule: The proofs in each of these four cases are verbatim identical to cases 4 through 7 in Lemma 4.8's proof. $\square$

8 and 9. The Proof for the Both the Bounded and Unbounded Cases where N_i is generated by the $\forall$-Elimination Rule: The combination of the inductive hypothesis and Lemma 4.7 implies the validity of Equation (47) by a completely trivial argument.

$$\text{VALUE}(t) \ < \ (k + \sqrt{y}) \cdot 2^{[\,\frac{1}{6}\beta_{i-1}\,]\,-\,2} \tag{47}$$

Since β_{i-1} represents the bit-length of the part of proof y that lies along the path σ_{i-1}, we have:

$$\beta_{i-1} \ < \ \text{Log}_2(y) \tag{48}$$

Recall our notation assumes $\frac{z}{x} = z$ when $x = 0$. Also, Equation (40) indicates $y^{\frac{3}{4}} < \frac{z}{x}$, and Lemma 5.3's hypothesis indicates $m = z - 1$ and $k = x$. In this context, Equations (47) and (48) easily imply

$$\text{VALUE}(t) \ \leq \ m \tag{49}$$

The remainder of the proofs of both Cases 8 and 9 for Lemma 5.3 is essentially verbatim identical to the comparable Cases 8 and 9 in Lemma 4.8's proof. This is because Equation (30) in Lemma 4.8's proof established a $\text{VALUE}(t) \leq m$ inequality, exactly analogous to Equation (49) above. The other parts of the proofs of Cases 8 and 9 (in the preceding section) were carefully written so that their discussion would apply to *any other ordered pair* (k, m) that satisfies the inequality $\text{VALUE}(t) \leq m$. Thus the verbatim same arguments will apply to Cases 8 and 9 of Lemma 5.3's proof as well. $\square$

10. The Proof for the Case where N_i is generated by the (Bounded version of the) $\exists$-Elimination Rule: Also, verbatim identical to Case 10 of Lemma 4.8's proof. $\square$

11. The Proof for the Case where N_i is generated by the (Unbounded version of the) $\exists$-Elimination Rule: Assuming some ancestor N_a of N_i stores the sentence $\exists v\, \Upsilon(v)$, the $\exists$-Elimination Rule allows N_i to indicate the sentence $\Upsilon(u)$ (where u denotes a new unused parameter symbol).

Define a **Founding Node** in a proof tree y to be a tree node that is either the tree-root or a node storing a proper axiom. Say the **Founding Antecedent** for an arbitrary node N_i is that particular *founding node* N_f such that N_i's sentence is produced by starting with the sentence in N_f and generating a finite string of deductions to construct N_i's sentence.

There are only two possible sentences, lying in a founding node of one of $\text{IS}^{\lambda}(A)$'s proof-trees, which can contain an *unbounded* existential quantifier. These are:

A. The proper axiom (given in Equation (17)) which indicates Addition is a total function.

B. The sentence stored in the proof's root.

The proof of Case 11 will be divided into the two sub-cases where the node N_i's Founding Antecedent is of Type-A and of Type-B. In both cases, the node N_i, generated by the ∃-Elimination Rule, will be shown to satisfy the (k, m)-Affirmative Condition.

Justification for the Sub-Case (A). Below are Equation (17)'s "Secondary" and "Tertiary" deductions:

$$\exists z \quad t_1 \;=\; z - t_2 \tag{50}$$

$$t_1 \;=\; u - t_2 \tag{51}$$

We need to inductively establish that that Equation (51) satisfies the $(k, m)-$Affirmative condition whenever its ancestor corresponding to (50) satisfies this condition. The proof of this sub-case is verbatim identical to the second paragraph in Case 11 of Lemma 4.8's proof.

Justification for the Sub-Case (B): This case had not appeared in Lemma 4.8's proof because Lemma 4.8's hypothesis implied that its tree-root contained the sentence "$0 \neq 1$" (which has no existential quantifiers). Since Lemma 5.3 allows for the more general case where any type of prenex* sentence can be stored in the tree-root, we must now consider certainly the possibility that the root contains an unbounded existential quantifier and the ∃-Elimination Rule derives the node N_i by eliminating this quantifier:

In Lemma 5.3, the root of the proof tree y is a prenex* sentence equivalent to $\neg\,\Phi$. We may assume $\neg\,(\Phi_z^x)$ is valid (because Lemma 5.3's hypothesis indicates Equation (40) is true).

From the fact that $\neg\,(\Phi_z^x)$ is valid, one can easily construct a valuation where each parameter u, generated by the root's existential quantifiers, will satisfy $\text{VALUE}(u) \leq x$ (see footnote [11] for this construction). We can once again assume that $\beta_i \geq 12$ (because merely the single sentence stored in the root shall require at least 12 bits for encoding it, under all conventional encoding methods, including the particular encoding method employed in Appendix B). Since the node in our present case is a sentence "$\Upsilon(u)$" that was deduced from an ancestor node "$\exists v\,\Upsilon(v)$", the inductive hypothesis allows us to presume that there exists a valuation where $\overbrace{\exists v\,\Upsilon(v) \;\Downarrow\; m}$ is true. Also Equation (40) trivially implies[12] $x \leq z - 1$. Since Lemma 5.3 discusses

[11] Normally the superscript x in Φ_z^x designates a bound on Φ's universal quantifiers. However, since the root of our semantic tableaux proofs consists of $\neg\,\Phi$, written in prenex* form, and because Equation (40) had indicated that $\neg\,[\Phi_z^x]$ is true, the positions of universal and existential quantifiers *reverses*, with x designating a bound on the root's existential quantifiers instead. The validity of $\neg\,[\Phi_z^x]$ thus implies each parameter u replacing Φ's existential quantifiers can have $\text{VALUE}(u) \leq x$ under a valuation we are able to construct.

[12] The integer y representing the Gödel number of a proof is certainly larger than say 4 under Appendix B's Gödel encoding scheme. From Equation (40), this trivially implies $x \leq z - 1$.

a $(k,m)-$Affirmative Branch where the constants $k = x$ and $m = z - 1$, it is easy to infer from the algebraic identities listed in this paragraph that that there exists a valuation satisfying Equations (52) and (53).

$$\overbrace{\Upsilon(u) \Downarrow m} \tag{52}$$

$$\text{VALUE}(u) \; < \; (\, k + \sqrt{y}\,) \cdot 2^{[\,\frac{1}{6}\beta_i\,]-2} \tag{53}$$

These two equations show that the node N_i satisfies Parts-I and II (respectively) of the definition of a $(k,m)-$Affirmative Branch. $\square$

Remark 5.4. It is significant that Theorem 5.1 was proven for a parameter $\lambda = \frac{3}{4} < 1$. The next section will need that $\lambda < 1$ to establish $\text{IS}^{\lambda}(A)$'s "Tangibility Reflection Principle".

Remark 5.5. Many of our upper bounds were needlessly relaxed so that we could avoid cluttering the proofs with added notation. It is for this reason that a tighter proof of Theorem 5.1 could establish the theorem for $\lambda = .01$ (and probably for any $\lambda \geq 10^{-4}$ or even yet smaller).

Remark 5.6 It is easy to generalize Theorem 5.1 so that $\text{IS}^{\lambda}(A)$'s Group-3 axioms can use any cut-free proof method, such as Herbrand Deduction, Resolution or the Cut-Free Sequent Calculus.

6 The Tangibility Reflection Principle and Its Limitations

Recall that $\text{TangRoot}_k(x)$ denotes the formula " $\exists v \;\; x < v^{1/k}$ ", and that $\text{IS}^{\lambda}(A)$'s TangRoot Reflection Principle for the sentence Ψ was defined as the assertion:

$$\forall x \;\; \{ \;\; [\;\; \exists \; y \;\; \text{SemPrf}_{\text{IS}^{\lambda}(\text{A})} (\ulcorner \Psi \urcorner \, , \; y \;) \; \wedge \; \text{TangRoot}_k(x) \;\;] \;\; \supset \Psi^x \;\; \} \tag{54}$$

We will have two goals in this section. Theorems 6.1 and 6.2 will establish that for every sentence Ψ, the axiom system $\text{IS}^{\lambda}(A)$ can verify its TangRoot_k Reflection Principle for any fixed $k > \frac{1}{1-\lambda}$. The second part of this section will explain why no real analog of the Tangibility Reflection Principle is available for axiom systems recognizing Multiplication as a total function.

Theorem 6.1 Let (λ, k) denote two constants satisfying $0 < \lambda < 1$ and $k > \frac{1}{1-\lambda}$. Assume that the axiom system A is strong enough to prove (see footnote[13]) the following Π_1^- theorem:

$$\forall x \, \forall y \, \forall v \;\; \{ \;\; x < v^{\frac{1}{k}} \;\; \supset \;\; y^{\lambda} < \frac{\text{MAX}(v,y)}{x} \;\; \} \tag{55}$$

Then for any prenex* sentence Ψ, there will exist a **"HILBERT-STYLE"** proof from $\text{IS}^{\lambda}(A)$'s proper axioms of Equation (54)'s Reflection Principle for Ψ.

[13]The intuitive reason Equation (55) is true is simply that the $k > \frac{1}{1-\lambda}$ inequality implies that $y^{\lambda} < \frac{y}{x}$ when $x < y^{1/k}$, and that $y^{\lambda} < \frac{v}{x}$ when $y^{1/k} \leq x < v^{1/k}$.

Proof. Below is the generic form of $\mathrm{IS}^{\lambda}(A)$'s Group-3 axiom:

$$\forall x\, \forall y\, \forall z \quad \{ \quad \mathrm{SemPrf}_{\mathrm{IS}^{\lambda}(A)} (\ulcorner \Psi \urcorner , \ y \ ) \ \wedge \ y^{\lambda} < \frac{z}{x} \quad \supset \quad \Psi^{x}_{z} \ \} \tag{56}$$

From (56), a Hilbert proof system can immediately deduce (57) (essentially because (57) is identical to (56) except that $\mathrm{MAX}(v, y)$ is substituted in the place of z).

$$\forall x\, \forall y\, \forall v \quad \{ \quad \mathrm{SemPrf}_{\mathrm{IS}^{\lambda}(A)} (\ulcorner \Psi \urcorner , \ y \ ) \ \wedge \ y^{\lambda} < \frac{\mathrm{MAX}(v, y)}{x} \quad \supset \quad \Psi^{x}_{\mathrm{MAX}(v,y)} \ \} \tag{57}$$

From the combination of (55) and (57), a Hilbert proof system can immediately deduce (58).

$$\forall x\, \forall y\, \forall v \quad \{ \quad \mathrm{SemPrf}_{\mathrm{IS}^{\lambda}(A)} (\ulcorner \Psi \urcorner , \ y \ ) \ \wedge \ x < v^{\frac{1}{k}} \quad \supset \quad \Psi^{x}_{\mathrm{MAX}(v,y)} \ \} \tag{58}$$

Moreover since Lemma 3.1.C can be proven by $\mathrm{IS}^{\lambda}(A)$, it can verify that $\Psi^{x}_{\mathrm{MAX}(v,y)} \supset \Psi^{x}$. Hence using the preceding identity and (58), Line (59) is easily derived:

$$\forall x\, \forall y\, \forall v \quad \{ \quad \mathrm{SemPrf}_{\mathrm{IS}^{\lambda}(A)} (\ulcorner \Psi \urcorner , \ y \ ) \ \wedge \ x < v^{\frac{1}{k}} \quad \supset \quad \Psi^{x} \ \} \tag{59}$$

Lastly, Line (54) can be trivially derived from Line (59) because they are obviously equivalent. □

Gentzen's Cut Elimination Theorem [11, 13, 35, 37] implies that every Hilbert-style proof can be transformed into a Semantic Tableaux proof of the same theorem. Hence, Theorem 6.1 immediately yields:

Corollary 6.2. For any prenex* sentence Ψ, the hypothesis of Theorem 6.1 also implies there exists a *Semantic Tableaux* proof from $\mathrm{IS}^{\lambda}(A)$ of Equation (54)'s TangRoot Reflection Principle for Ψ.

Remark 6.3. This paragraph will define a slightly stronger version of the Tangibility Reflection Principle. Let Ψ_{μ} denote a prenex* formula whose only free variable is μ , and $\Psi_{\mu}(k)$ be a sentence which replaces each occurrence of μ with the constant k . Also $\ulcorner \Psi_{\mu}(k) \urcorner$ will denote $\Psi_{\mu}(k)$'s Gödel number, and $\Psi_{\mu}(k)^{x}$ will denote a sentence identical to $\Psi_{\mu}(k)$ except that each unbounded universal quantifier from $\Psi_{\mu}(k)$ is now bounded by x. Then an axiom system α will be said to support the **Uniform Tangibility Reflection Principle** if it can prove (60) for each prenex* formula Ψ_{μ}

$$\forall v\, \forall x \quad \{ \quad [\quad \exists \ y \quad \mathrm{Prf}_{\alpha} (\ulcorner \Psi_{\mu}(\dot{v}) \urcorner , \ y \ ) \wedge \mathrm{Tangible}(x) \quad] \quad \supset \Psi_{\mu}(\dot{v})^{x} \quad \} \tag{60}$$

It is possible to develop axiom systems, similar to $\mathrm{ISREF}(A)$ and $\mathrm{IS}^{\lambda}(A)$, which support the Uniform Tangibility Reflection Principle. These two axiom systems will be called $\mathrm{ISREF}_{U}(A)$ and $\mathrm{IS}^{\lambda}_{U}(A)$. They will have the identical Group-zero, Group-1 and Group-2 axioms as $\mathrm{ISREF}(A)$ and $\mathrm{IS}^{\lambda}(A)$. They will differ only by having an uniformized version of their Group-3 axiom schema. This topic is mentioned because it is philosophically curious that one can achieve the added level of self-knowledge specified in (60). The proofs of the consistency of $\mathrm{ISREF}_{U}(A)$ and $\mathrm{IS}^{\lambda}_{U}(A)$ are actually quite similar to the analysis of $\mathrm{ISREF}(A)$ and $\mathrm{IS}^{\lambda}(A)$. We did not provide them in this article because our proofs would then be cluttered with one extra level of notation.

CLARIFYING COMMENTS ABOUT MULTIPLICATION: Some readers may wonder whether an analog of $\mathrm{IS}^{\lambda}(A)$ could recognize Multiplication as a total function. In particular, $\mathrm{IS}^{\lambda}(A)$'s Group-3 axiom (9) required that the inequality $y^{\lambda} < \frac{z}{x}$ separate the magnitudes of y from z. It is natural to inquire whether an alternate "ISM(A)" system could view Multiplication as a total function, if its Group-3 axiom schema employed the canonical form (61) instead. Note (61)'s inequality $y \cdot \mathrm{Log}_2(x+2) < \mathrm{Log}_2(z)$ forces a wider separation between the magnitudes of y and z than does (9)'s analogous inequality $y^{\lambda} < \frac{z}{x}$.

$$\forall x \, \forall y \, \forall z \quad \{ \quad \mathrm{SemPrf}_{\mathrm{ISM(A)}} (\ulcorner \Psi \urcorner , \, y) \quad \wedge \quad y \cdot \mathrm{Log}_2(x+2) < \mathrm{Log}_2(z) \quad \supset \quad \Psi_z^x \quad \} \qquad (61)$$

The answer to the preceding question is quite surprisingly both affirmative and negative !

The positive aspect will be that the ISM($\bullet$) mapping does indeed possess consistency preservation properties analogous to $\mathrm{IS}^{\lambda}(\bullet)$. Thus, if A is regularly consistent then so will ISM(A) be consistent, by a routine generalization of Section 5's proof of Theorem 5.1.

However the crucial comparison is that $\mathrm{IS}^{\lambda}(A)$ (with the constant λ chosen to satisfy $\lambda < 1$) seems to have much more philosophical significance than ISM(A). This is because Theorem 6.2 showed that $\mathrm{IS}^{\lambda}(A)$ could prove its TangRoot Reflection Principles. Also, Section 1's discussion of Equation (4) and (5) explained how $\mathrm{IS}^{\lambda}(A)$ could deduce from its Tangibility Reflection the theorem that

$+++$ " I am unable to produce a Semantic Tableaux proof of 0=1 "

However, Equation (61) is too weak for ISM(A) to prove analogously: " $\forall y \neg \mathrm{SemPrf}_{ISM(A)}(\ulcorner 0 = 1 \urcorner, y)$ ". The reason ISM(A) is unable to deduce this sentence is because the gap between the sizes of y and z is simply too large in Equation (61) (see footnote[14]).

Thus oddly, ISM(A) and $\mathrm{IS}^{\lambda}(A)$ have quite different *philosophical and epistemological* implications, although the strictly *formalistic* mathematical proofs of their consistency are virtually *identical.*

Nor is any other axiom system known which recognizes Multiplication as a total function and which can prove an analog of Equation (54)'s Reflection Principle, using either TangRoot or *any other slower growing* Tangibility formula. A short intuitive explanation of why all the known Consistency-Preservation proofs do collapse when Multiplication is present was given in Remark 4.5. The theorems in the next section will discuss formalisms that are incompatible with the recognition of Multiplication as a total function.

[14] Using $x = 0$ and $\Psi =$ "0=1" in Equation (61), ISM(A) can prove $\forall y \, [\exists z \;\; y < \mathrm{Log}_2(z)] \supset \neg \mathrm{SemPrf}_{\mathrm{ISM(A)}}(\ulcorner 0 = 1 \urcorner, y)$. However, the gap between the sizes of y and z in (61) is too large to also imply $\forall y \, \neg \mathrm{SemPrf}_{\mathrm{ISM(A)}} (\ulcorner 0 = 1 \urcorner , \, y)$

7 Three New Variations of the Second Incompleteness Theorem:

This section will have two goals. One will be to prove the new variation of Gödel's Second Incompleteness Theorem that was mentioned by Item C of Section 1. This version of the Second Incompleteness Theorem will explain why it is infeasible for any version of a self-justifying axiom system to employ Equation (1)'s "Canonical Reflection Principle". It will thus explain why our ISREF(A) and IS$^{\lambda}$(A) axiom systems had instead relied on Equation (2)'s weaker "Tangibility Reflection Principle".

Our second goal will be to prove that certain types of generalizations of the IS$^{\lambda}$(A) axiom system become infeasible when they recognize Multiplication as a total function.

Our theorems will be applicable to all the conventional methods of deduction, including Hilbert-style proofs, Herbrand-style proofs, Gentzen-style sequent calculus proofs, Semantic Tableaux proofs, Resolution proofs, etc. They will also be sufficiently general to apply to possible future methods, which unlike the preceding methods, actually do not need to be *"fully" sound and complete.* In particular, let D denote a method of deduction, α denote a set of proper axioms, and let " $\alpha \vdash_D \phi$ " denote that there exists a proof from α, using deduction-method D, of the theorem-sentence ϕ. We will say that the deduction-method D is $\alpha-$**Sound** iff $\alpha \vdash_D \phi$ implies that ϕ is true in every model of α. Similarly, define the deduction-method D to be $\alpha-$**Complete** iff $\alpha \vdash_D \phi$ holds whenever ϕ is true in every model of α.

Conventional methods of deduction D are obviously $\alpha-$sound and $\alpha-$complete, for all axiom systems α . For the pleasure of added generality, the theorems in this section will also pertain to deductive methods which are merely sound and complete *locally to* α .

Lemma 7.1. If α denotes a set of proper axioms and D denotes a deduction method that is both $\alpha-$sound and $\alpha-$complete then

A. The combination of $\alpha \vdash_D \phi$, $\alpha \vdash_D \psi$, and $\alpha \vdash_D \phi \wedge \psi \supset \varphi$ implies $\alpha \vdash_D \varphi$.

B. The combination of $\alpha \vdash_D \phi$, $\alpha \vdash_D \psi$, and the assumption that " $\phi \wedge \psi \supset \varphi$ " holds in every model $\mathcal{M}$ of α implies $\alpha \vdash_D \varphi$.

Proof. We will prove only Claim A, since both claims have essentially identical proofs. From the fact that D is $\alpha-$sound and from Claim A's hypothesis, it follows that each of the sentences ϕ, ψ and $\phi \wedge \psi \supset \varphi$ hold in every model $\mathcal{M}$ of α. Hence, so does φ hold in every model $\mathcal{M}$ of α. Since D is $\alpha-$complete, the latter implies that $\alpha \vdash_D \varphi$. □

Comment: Let p, q and r denote proofs of the sentences ϕ, ψ and $\phi \wedge \psi \supset \varphi$ (above). If D designates a Hilbert-style proof method, then it is well known how to construct a proof s of φ by essentially

concatenating together the three initial proofs p, q and r. However, it is also known that for cut free proof methods, such as Semantic Tableaux, the construction of s is more complicated: It could then correspond to a bit string more than exponentially larger than the combined lengths of the three initial proofs p, q and r. The significance of Lemma 7.1 is that it asserts that s always exists (although it may be exceedingly difficult to construct). We will use Lemma 7.1 often in the proofs of Theorems 7.2, 7.3 and 7.4

Notation. The symbol **PAX** will denote the trivial extension of Peano Arithmetic whose function symbols include the seven Grounding functions, in addition to Multiplication and Addition. The formula $\mathrm{Prf}_{\alpha}^{D}(x, y)$ will designate that y is a proof of the sentence x from the axiom system α using the deduction method D. The system α's "Canonical Reflection Principle" for the sentence Υ is the assertion:

$$\forall y \quad \{ \quad \mathrm{Prf}_{\alpha}^{D} (\ulcorner \Upsilon \urcorner , y) \quad \supset \quad \Upsilon \quad \} \tag{62}$$

Theorem 7.2. Suppose α is a consistent axiom system, D is an $\alpha-$sound and $\alpha-$complete method of deduction, $\mathrm{Prf}_{\alpha}^{D}(x, y)$ is a Δ_0^- formula, and α can verify all PAX's Π_1^- theorems. Then α cannot prove the validity of the "Canonical" Reflection Principle (62) for every Π_1^- sentence Υ.

Comment. The above result is different from Löb's Theorem [3, 13, 22, 23] chiefly because we do not assume that α recognizes either Addition or Multiplication as total functions.

Proof of Theorem 7.2. We will first show how to use a fix-point construction to encode a Π_1^- sentence Θ , whose translation into English is roughly that:

$* *$ There is no proof of this sentence from the axiom system α.

Let $\mathrm{SUBST}(g, h)$ denote Gödel's classic substitution relation, defined below:

$\mathrm{SUBST}(g, h)$ = The integer g is an encoding of a formula, and h encodes a sentence identical to g, except that all free variables in g are replaced with a constant, whose value equals g.

Then Θ can be defined as being simply the sentence $\Gamma(\bar{n})$, where $\Gamma(g)$ denotes the formula given in Equation (63) and $\bar{n}$ denotes $\Gamma(g)$'s Gödel number.

$$\forall x \; \forall h \leq x \quad \mathrm{SUBST}(g, h) \quad \supset \quad \neg \, \mathrm{Prf}_{\alpha}^{D} \, (h , x) \tag{63}$$

We may also assume that $\mathrm{SUBST}(g, h)$ can be encoded as a Δ_0^- formula, since Appendix C will show how this can be done (i.e. see its Theorems C.1.12 and C.2). Since the hypothesis of Theorem 7.2 indicated that $\mathrm{Prf}_{\alpha}^{D} \, (h , x)$ was also a Δ_0^- formula, these facts imply Θ is a Π_1^- sentence.

Now suppose for the sake of contradiction that α could prove the validity of its canonical reflection principle (62) for all Π_1^- sentences. Then since Θ is Π_1^-, we have:

$$\alpha \vdash_D \quad \forall y \quad \{ \quad \mathrm{Prf}_\alpha^D (\ulcorner \Theta \urcorner , y) \quad \supset \quad \Theta \ \} \tag{64}$$

Applying a degenerate form of Lemma 7.1B to Equation (64), we get:

$$\alpha \vdash_D \quad \{ \ \forall y \ \neg \ \mathrm{Prf}_\alpha^D (\ulcorner \Theta \urcorner , y) \ \} \ \vee \ \Theta \tag{65}$$

We will complete the proof of Theorem 7.2 by exploring the diagonalization properties of Θ .

Since Θ is defined to be the mathematical sentence which corresponds to the English sentence **, it is evident that Equation (66) (below) is clearly valid.

$$\Theta \quad \Leftrightarrow \quad \{ \ \forall y \ \neg \ \mathrm{Prf}_\alpha^D (\ulcorner \Theta \urcorner , y) \ \} \tag{66}$$

However, we need slightly more than this fact because we must establish that the *weak axiom system* α can verify (66). The latter is easy to justify because Θ's diagonalization construction implies that Θ's Gödel number is the *unique integer* satisfying $\mathrm{SUBST}(\bar{n} , \ulcorner \Theta \urcorner)$ (where $\bar{n}$ was defined as the Gödel number of Equation (63)). The key point is that α *can prove* this fact about $\ulcorner \Theta \urcorner$'s Uniqueness. The reason α can prove $\ulcorner \Theta \urcorner$ is the unique integer satisfying $\mathrm{SUBST}(\bar{n} , \ulcorner \Theta \urcorner)$ is that it is provable from PAX as a Π_1^- theorem: It is thus accessible to α (by Theorem 7.2's hypothesis). Once α has verified $\ulcorner \Theta \urcorner$ is the unique number satisfying $\mathrm{SUBST}(\bar{n} , \ulcorner \Theta \urcorner)$, it can easily prove theorem (66). Hence, this paragraph has established

$$\alpha \vdash_D \quad \Theta \ \Leftrightarrow \ \{ \ \forall y \ \neg \ \mathrm{Prf}_\alpha^D (\ulcorner \Theta \urcorner , y) \ \} \tag{67}$$

Applying Lemma 7.1B to the combination of Equations (65) and (67), we get:

$$\alpha \ \vdash_D \quad \Theta \tag{68}$$

From (68), we know that there exists some integer $\bar{m}$ such that $\mathrm{Prf}_\alpha^D (\ulcorner \Theta \urcorner , \bar{m})$ is true. Moreover because α can prove all PAX's Π_1^- theorems, it obviously has a capacity to prove every Δ_0^- sentence which is true. Hence, we conclude that:

$$\alpha \vdash_D \quad \mathrm{Prf}_\alpha^D (\ulcorner \Theta \urcorner , m) \tag{69}$$

From (67), (69) and another application of Lemma 7.1B, we get:

$$\alpha \ \vdash_D \quad \neg \Theta \tag{70}$$

Hence the combination of (68) and (70) shows that the axiom system α is inconsistent. This completes our proof of Theorem 7.2 because it shows that it is infeasible for α to be simultaneously consistent and to prove the validity of its canonical reflection principle (62) for all Π_1^- sentences. $\square$

Theorem 7.3. Define $\mathrm{XIS}^{\lambda}(\mathrm{PAX})$ to be an axiom system identical to $\mathrm{IS}^{\lambda}(\mathrm{PAX})$ except that

A. $\mathrm{XIS}^{\lambda}(\mathrm{PAX})$'s Group-1 axiom schema will recognize Multiplication as a total function.

B. The Group-3 axiom schema of $\mathrm{XIS}^{\lambda}(\mathrm{PAX})$ will obviously indicate $\mathrm{XIS}^{\lambda}(\mathrm{PAX})$'s reflection properties (rather than $\mathrm{IS}^{\lambda}(\mathrm{PAX})$'s reflection principle). Thus for each prenex* sentence Θ, there will be a corresponding Group-3 axiom of the form:

$$\forall x\ \forall y\ \forall z\ \{\ [\mathrm{SemPrf}_{\mathrm{XIS}^{\lambda}(\mathrm{PAX})}\ (\ \ulcorner \Theta \urcorner\ ,\ y\ )\ \wedge\ y^{\lambda} < \frac{z}{x}\ ]\ \supset\ \Theta^{x}\ \} \tag{71}$$

Then for all values of the parameter λ, the axiom system $\mathrm{XIS}^{\lambda}(\mathrm{PAX})$ is inconsistent.

Proof. Since the axiom system $\mathrm{XIS}^{\lambda}(\mathrm{PAX})$ recognizes Multiplication as a total function, Equation (72) must be valid *for any fixed constant constant* $\lambda > 0$ *:*

$$\mathrm{XIS}^{\lambda}(\mathrm{PAX})\ \vdash\ \ \forall x\,\forall y\,\exists z\ \ y^{\lambda} < \frac{z}{x} \tag{72}$$

Using Equation (72) and the fact that (71) is an axiom of $\mathrm{XIS}^{\lambda}(\mathrm{PAX})$, Lemma 7.1B implies:

$$\mathrm{XIS}^{\lambda}(\mathrm{PAX})\ \vdash\ \ \forall x\ \forall y\ \{\ \mathrm{SemPrf}_{\mathrm{XIS}^{\lambda}(\mathrm{PAX})}\ (\ \ulcorner \Theta \urcorner\ ,\ y\ )\ \supset\ \Theta^{x}\ \} \tag{73}$$

Also for any Π_1^- sentence Θ, an absolutely trivial argument shows:

$$\mathrm{XIS}^{\lambda}(\mathrm{PAX})\ \vdash\ \ \{\ \forall x\ \ \Theta^{x}\ \}\ \supset\ \Theta \tag{74}$$

Applying Lemma 7.1.B to Equations (73) and (74), we get:

$$\mathrm{XIS}^{\lambda}(\mathrm{PAX})\ \vdash\ \ \forall y\ \{\ \mathrm{SemPrf}_{\mathrm{XIS}^{\lambda}(\mathrm{PAX})}\ (\ \ulcorner \Theta \urcorner\ ,\ y\ )\ \supset\ \Theta\ \} \tag{75}$$

Since the above holds for all Π_1^- sentences Θ, the combination of Theorem 7.2 and Eq. (75) imply $\mathrm{XIS}^{\lambda}(\mathrm{PAX})$ is inconsistent. (This is because Theorem 7.2 does *not* allow a *consistent* axiom system $\mathrm{XIS}^{\lambda}(\mathrm{PAX})$ to have the capacity indicated by (75) simultaneously for all Π_1^- sentences Θ .) $\square$

Comment. Theorem 7.3 clearly shows no hybrid of Robinson's System Q with $\mathrm{IS}^{\lambda}(A)$ is feasible that both recognizes Multiplication as a total function and can verify its TangRoot Reflection Principle. Yet although "hybrids" between Q and $\mathrm{IS}^{\lambda}(A)$ are infeasible, some very interesting "interfaces" between $\mathrm{IS}^{\lambda}(A)$ and any A extending Q are made possible by the theory of Definable Cuts [12, 13, 17, 18, 24, 26, 28, 29, 38, 39] See footnote 2 of Section 2 for an example of a surprising class of "interfaces" that can be formally constructed.

Theorem 7.4. Suppose α is a consistent axiom system, it **recognizes Multiplication** as a Total Function, D is an $\alpha-$sound and $\alpha-$complete method of deduction, $\mathrm{Prf}_{\alpha}^{D}(x,y)$ is a Δ_0^- formula, α can verify all PAX's Π_1^- theorems, and that for every Δ_0^- formula $\phi(x)$ the system α can also prove (see footnote[15]) the sentence (76) for some fixed constant k .

$$\forall p \ \forall a \ \forall b \ \{ \quad [\ b \geq (p \cdot a)^k \ \wedge \ \mathrm{Prf}_{\alpha}^{D}(\ \ulcorner \forall x \phi(x) \urcorner \ , \ p \ ) \ ] \quad \supset \quad \exists q < b \ \ \mathrm{Prf}_{\alpha}^{D}(\ \ulcorner \phi(\dot{a}) \urcorner \ , \ q \ ) \ \} \tag{76}$$

Then α *cannot* prove the validity of the theorem (77) (below) for all Δ_0^- formulae $\phi(x)$, simultaneously.

$$\forall \ a \quad \{ \ [\ \exists \ q \ \ \mathrm{Prf}_{\alpha}^{D} \ (\ \ulcorner \phi(\dot{a}) \urcorner \ , \ q \ ) \ ] \quad \supset \quad \phi(a) \quad \} \tag{77}$$

(Equation (77) is often called [9, 33] $\phi(x)$'s "Second Uniform" Reflection Principle.)

Proof Sketch: We will only sketch Theorem 7.4's proof because it is very similar to Theorem 7.3's proof. Applying Lemma 7.1.B to the combination of the fact that α recognizes Multiplication as a total function and that it can prove (76), implies that α can also prove:

$$\forall p \ \forall a \ \exists q \ \ \{ \ \mathrm{Prf}_{\alpha}^{D} \ (\ \ulcorner \forall x \phi(x) \urcorner \ , \ p \ ) \quad \supset \quad \mathrm{Prf}_{\alpha}^{D} \ (\ \ulcorner \phi(\dot{a}) \urcorner \ , \ q \ ) \ \} \tag{78}$$

We will prove Theorem 7.4 by using a proof-by-contradiction. For the sake of establishing a contradiction, let us assume α can prove the theorem (77) for every Δ_0^- formulae $\phi(x)$. Applying Lemma 7.1.B to the combination of α's ability to prove (77) and (78), implies α can also prove "$\forall p \{ \mathrm{Prf}_{\alpha}^{D}(\ulcorner \forall x \phi(x) \urcorner , p) \supset \forall x \phi(x) \}$", for every Π_1^- sentence "$\forall x \phi(x)$". This implies α is *inconsistent* (because Theorem 7.2 precludes a consistent α from having such reflection capacities).

Hence, it is impossible for a consistent α to prove all sentences of the form (77). $\square$

Comment: Part of the reason that Theorem 7.4 is interesting is that its negative result contrasts sharply with the uniform reflection principle which Remark 6.3 associated with ISREF(A) and IS$^{\lambda}$(A). It also should be noted that during the period while this article was being refereed, we developed yet another version of the Second Incompleteness Theorem. It concerns Remark 5.5's ISMULT formalism. Our theorem shows that for every α extending PAX, the axiom system ISMULT(α) is inconsistent. A 16-page description of this theorem was published by us recently in the Semantic Tableaux 2000 Conference Proceedings, i.e. see [44].

8 Conclusion

There are several additional aspects of self-justifying axiom systems which we have postponed until a future paper. For example, we have represented a "proof" as a string of bits associated with some integer. An

[15] A warning is that for some unusual semantic tableaux proof systems, the statement (76) is fallacious for all k because there is inadequate proof compression. However, virtually all Semantic Tableaux systems α will satisfy (76) for at least some *fixed* k , provided that for each natural number i they represent i with a constant symbol $\overline{C_i}$ and contain the *very trivial* formal statement " $\exists x = \overline{C_i}$ " as an *axiom* (rather than as a theorem).

alternate definition would allow a proof to correspond to the bit string produced by a computer whose input and permitted running times are specified by arbitrary integers. (This notion can be formalized with the use of Appendix B's Turing functions). In this context, there are interesting generalizations of IS(A), ISREF(A) and IS$^{\lambda}$(A), whose Group-3 axioms use "*I am consistent*" statements where the " I " refers to a computer able to generate in N units of time a bit-string of roughly N bits. (These strings are much longer than the integer N itself, which contains only Log(N) bits). Such computable self-verifying systems are very subtle because they can become inconsistent if their "*I am consistent*" statement is excessively strong. (The difficulty is that excessive strength can cause such systems α to violate Part-ii of Section 1's definition of self-justification.) For example, a computable version of IS(A) would be feasible if it recognizes Successor as a total function and retains the Group-zero axiom that Predecessor($\overline{n}$) = $\overline{n-1}$. However, the computable variants of IS(A) will be excessively strong and inconsistent if they either recognize Addition as a total function or retain the Group-zero axiom that $\overline{2n} - \overline{n} = \overline{n}$. (Moreover, the axiom that Successor is a total function must be removed from the computable version of IS(A) if one wishes to revise its Group-3 schema to use Hilbert-style proofs rather than Semantic Tableaux structures.)

There are also some interesting links between the P≠NP open question and the computable versions of IS(A) discussed in [42]. However frankly, we are unsure whether any of the new open questions posed in [42] are any easier to solve than the original P≠NP open problem.

The chief reason for our interest in the ISREF(A) and IS$^{\lambda}$(A) axiom systems is that they seem to offer some insights into a version of the Liar's Paradox, raised by Gödel's Incompleteness Theorem. In particular, they seem to provide an interesting partial answer to the paradoxical question below:

* How do Human Beings manage to muster the physical energy and psychological desire to think (and prove theorems) when the various generalizations of Gödel's Incompleteness Theorem assert that no reasonable conventional axiom system can confidently assume its own consistency?

We can offer no full answer to the preceding question because paradoxes never have a full resolution. However, our partial answer to the Question $*$ is that a Thinking Being can assume that if he proves Ψ, then Ψ is valid when it is restricted to numbers of "*reasonable size*". In particular, each of TangPred(x) , TangRoot(x) and TangDiv(x) can formalize what is meant by a number of "*reasonable size*" in Equation (2).

Since none of our axiom systems will recognize Multiplication as a total function and since some will not recognize Addition as a total function, these axiom systems will clearly be awkward in some serious respects. However, what is more curious about the Tangibility Reflection Principle can perhaps be seen when it is examined from the exact opposite perspective. This is that Theorem 7.2 proves that *even after one drops the axioms* that Addition and Multiplication are total functions, an axiom system α will *remain inconsistent* if

it includes an ability to verify Equation (1)'s Canonical Reflection Principle.

From this context, Equation (2)'s Tangibility Reflection Principle becomes especially enticing because it shows that close approximations to the infeasible Equation (1) are plausible.

The Introduction Chapter of this article had forewarned our readers that it was extremely difficult to uniformly compare ISREF(A) and IS$^{\lambda}$(A) to the prior results discussed by [20, 21, 24, 28, 36, 38]. In particular, Section 1 listed seven criteria where one could attempt to compare the self-verifying abilities of sundry axiom systems. It pointed out that the prior literature would be roughly preferable to ISREF(A) and IS$^{\lambda}$(A) from the perspectives of Criteria I through IV. However, ISREF(A) and IS$^{\lambda}$(A) would add some new insights from the measures of perspectives V, VI and VII.

The salient point is that the generalizations of the Second Incompleteness Theorem, by Pudlák[28], Solovay[34] and our Theorems 7.2, 7.3 and 7.4, show that it is impossible for a self-verifying system to be optimal from all of perspectives (I)-(VII) *simultaneously !* Thus, there are inherent trade-offs, where new insights arise only when considering axiom systems with other new types of limitations.

Only from this perspective can ISREF(A) and IS$^{\lambda}$(A) be fairly judged. An axiom system is unquestionably less than ideal when it does not verify Addition and Multiplication are total functions. However by their very nature, paradoxes never have ideal solutions. The partial virtues of IS$^{\lambda}$(A) and ISREF(A) will be that their Tangibility Reflection Principles will offer at least a partial resolution for the Paradoxical Question $*$.

ACKNOWLEDGMENTS: I am extremely grateful to Robert Solovay for describing to me his unpublished version of the Second Incompleteness Theorem. (Its formal statement is given in Appendix A.) Solovay's theorem and Pudlák's analogous result[28], which Solovay also described to me, were very helpful in enabling me to avoid the temptation of trying to construct several seemingly natural hybrids of ISREF(A) and IS(A), that are indeed infeasible. I am very grateful for this.

I thank the University of Tel Aviv for hosting my sabbatical during 1992. It was during that period that I conceived the first version of a self-verifying axiom system, published in [40]. As a child of two parents who barely survived the Holocaust, I was glad that at least part of my research was done there.

I also thank the anonymous referee for having read two drafts of this paper and for having suggested many improvements in the presentation.

Appendix A: Solovay's Extension of Pudlák's Theorem

Robert Solovay (private communications, 1994) has observed there exists a finite set of Π_1^- theorems discussing the properties of merely subtraction and division, such that no consistent axiom system α can prove these theorems, recognize Successor as a total function, and prove no Hilbert proof of 0=1 exists from itself.

Numerous articles [13, 24, 26, 28, 29, 38] have credited an unpublished observation by Solovay for introducing the method of thinning Definable Cuts. The development of cut formulae for Robinson's System Q (that model $I\Sigma_0$) were due to Nelson [24] and to Wilkie-Paris [38, 39]. Pudlák [13, 28, 29] discovered the very crucial theorem that no extension of Q can prove its own Hilbert consistency. Solovay saw the very pretty theorem, described in this Appendix, as combining all these perspectives together.

The reason for our interest in Theorem A.2 is it will imply that all the seemingly natural hybrids of ISREF(A) and IS(A) are infeasible. In the interests of brevity, some aspects of this Appendix's proof will assume the reader is familiar with some of the prior works of Dimitracopoulos, Hájek, Krajícek, Nelson, Paris, Pudlák or Wilkie [13, 17, 18, 24, 26, 28, 29, 38] about Definable Cuts. This highly abbreviated format is reasonable because Theorem A.2 is never used in the main sections of our article. (Its sole purpose is to show, surprisingly, that all the seemingly natural hybrids of ISREF(A) and IS(A) are implausible.)

Let $\text{Derive}(x, y)$ denote a formula which indicates that the integer y represents a bit string that is some type of verification of the theorem whose Gödel number equals x . For example, $\text{Derive}(x, y)$ could be a formula indicating that y is a Hilbert-style proof of x . However, it also could represent many other types of deductive formalisms as well. Also, $\text{Der}(x)$ will be an abbreviation for " $\exists y\, \text{Derive}(x, y)$ ". Although " $\text{Der}(x)$ " can specify a very non-conventional method of deductive inference, the symbol " $\alpha \vdash \Upsilon$ " will retain its usual definition. It will specify that there exists a conventional Hilbert-style proof from the axiom system α of the theorem Υ . Before examining Solovay's and Pudlák's versions of the Second Incompleteness Theorem, it is useful to first consider the following generalization of the Hilbert-Bernays Theorem [14]:

Theorem A.1 Suppose α can prove all Peano Arithmetic's Π_1^- theorems. Suppose *for any two sentences* Φ and Ψ, the formula $\text{Der}(x)$ satisfies the following three conditions:

1. If $\alpha \vdash \Phi$ then $\alpha \vdash \text{Der}(\ulcorner \Phi \urcorner)$.
2. $\alpha \vdash \{ \text{Der}(\ulcorner \Phi \urcorner) \wedge \text{Der}(\ulcorner \Phi \supset \Psi \urcorner) \} \supset \text{Der}(\ulcorner \Psi \urcorner)$.
3. $\alpha \vdash \text{Der}(\ulcorner \Phi \urcorner) \supset \text{Der} \{ \ulcorner \text{Der}(\ulcorner \Phi \urcorner) \urcorner \}$.

Then if the axiom system α is consistent, it cannot prove $\neg \text{Der}(\ulcorner 0 = 1 \urcorner)$

Comment: Mendelson's textbook refers to the conditions (1)-(3) as the **"Hilbert-Bernays Derivability Conditions"**. Some other logicians have called them **"the Löb Conditions"** because Löb's Theorem

[22, 23] is the most famous theorem about these three conditions.

Justification of Theorem A.1. If the first sentence of Theorem A.1 had specified that α was an extension of Peano Arithmetic then Theorem A.1 would essentially correspond to Hilbert and Bernays' version of the Second Incompleteness Theorem. The hypothesis of Theorem A.1 differs from the classic Hilbert-Bernays theorem only by not requiring α to be an extension of Peano Arithmetic. However, it does require α to have an ability to prove all Peano Arithmetic's Π_1^- theorems. It turns out that this is actually the core fact that is needed by the classic proof of Hilbert-Bernays theorem.

In the interests of brevity, we will not give a formal proof of Theorem A.1 in this very short appendix. However, the footnote below[16] explains in detail the underlying reason why Theorem A.1's proof is fully identical to Hilbert and Bernays' well-known prior proof construction [14]. □

Notation: Since we will often allude to results by Hájek-Pudlák [13] and Wilkie-Paris [38], it should be pointed out that these two sources [13, 38] use slightly different notation conventions to define what is essentially the same concept. Let $\omega_0(x) = 2x$. For $i > 0$ define $\omega_{i+1}(0) = 0$ and $\omega_{i+1}(x) = 2^{\omega_i(\lceil Log_2(x+1) \rceil - 1)}$. Then Hájek-Pudlák [13] define Ω_i to be the sentence asserting that ω_{i+1} is a total function. Similarly, Ω_i^J is defined as the sentence asserting that ω_{i+1} is a total function locally within the restricted domain specified by the cut formula J . Let ϖ_0, ϖ_1, ϖ_2, ... denote the Wilkie-Paris [38] counterpart of the Hájek-Pudlák sequence ω_0, ω_1, ω_2, ... It is defined by the convention that $\varpi_0(x) = x$ and $\varpi_{i+1}(x) = x^{\varpi_i(\lceil Log_2(x+1) \rceil - 1)}$. Wilkie-Paris [38] define Ω_i as the sentence asserting that ϖ_i is a total function, and Ω_i^J to be the sentence asserting that ϖ_i is a total function locally within the cut formula J . It is well-known that $I\Sigma_0$ can prove that these two definitions of Ω_i are equivalent. Therefore the prior literature has found it mostly unnecessary to distinguish between the two slightly different notations.

Define an axiom system α to be **Successor-Regular** if it recognizes Successor as a total function, it can prove all Peano Arithmetic's Π_1^- theorems, $\text{HilbPrf}_\alpha(x, y)$ can be encoded as a Δ_0^- formula, and α is consistent. Since α can prove all Peano Arithmetic's Π_1^- theorems, we can use reasoning similar to the prior literature's treatment of $I\Sigma_0$ to justify that α can verify that the Hájek-Pudlák and Paris-Wilkie definitions of Ω_i are equivalent to each other when $i \geq 1$. However, the two definitions of Ω_0 are not equivalent

[16] Let $\text{Subst}(g, h)$ denote Gödel's classic substitution relation, which specifies that the integer g is an encoding of a formula, and h encodes a sentence identical to g, except that all free variables in g are replaced with a constant, whose value equals g. Let $\Theta(z)$ denote the formula " $\forall x \forall y$ $\text{Subst}(z, x)$ $\supset$ $\neg\text{Derive}(x, y)$ ", and N denote the Gödel number of this formula. It is trivial to see that the Π_1^- theorems of Peano Arithmetic are sufficient to prove that $\ulcorner \Theta(N) \urcorner$ is the unique integer satisfying $\text{Subst}(\, N \,, \ulcorner \Theta(N) \urcorner \,)$. This immediately implies that α can prove that $\Theta(N) \Leftrightarrow \neg\text{Der}(\ulcorner \Theta(N) \urcorner)$. It turns out that the preceding fixed point identity is the only aspect of the proof of the Hilbert-Bernays Theorem that needs Peano Arithmetic for justifying it. In other words, the remainder of the proof of the Hilbert-Bernays Theorem rests solely on the fact that α has the logical strength indicated by Theorem A.1's hypothesis (i.e. that α supports conditions (1)-(3)). It is for this reason that Theorem A.1's proof is basically identical to the proof of the Hilbert-Bernays theorem.

under some Successor-Regular axiom systems α , which do not recognize Addition and Multiplication as total functions. Moreover since Hájek-Pudlák use a different subscript convention where ω_i is roughly comparable to ϖ_{i-1} , its notation can also define Ω_i in the special degenerate case where $i = -1$. We will therefore use the Hájek-Pudlák notation in the special cases where Ω_i has a subscript $i = 0$ or $i = -1$.

The prior literature [13, 17, 18, 24, 26, 28, 29, 38, 39] has illustrated many different variants of the method of "thinning cuts". In general, most arithmetic axiom systems A can define a sequence of Definable Cuts $J_0,\ J_1,\ J_2,\ J_3$ .... such that A can prove:

1. the validity of the sentence " $\Omega_{i-1}^{J_i}$ ",
2. that " J_i is a Definable Cut",
3. and that " $\forall x\ J_i(x) \supset J_k(x)$ " for any two fixed constants satisfying $k < i$.

For example, if α is a Successor-Regular axiom system, then $J_0,\ J_1,\ J_2,\ J_3$ can be defined as:

$$J_0(v) \quad \equiv_{\rm df} \quad \forall u\ \exists z\ \ v = z - u \tag{79}$$

$$J_1(v) \quad \equiv_{\rm df} \quad J_0(v)\ \wedge\ \forall u\ \exists z\ \{\ J_0(u)\ \supset\ [J_0(z)\ \wedge\ v = \frac{z}{u}\ ]\ \} \tag{80}$$

$$J_2(v) \quad \equiv_{\rm df} \quad J_1(v)\ \wedge\ \forall u\ \exists z\ \{\ J_1(u)\ \supset\ [J_1(z)\ \wedge\ {\rm Log}(v) = \frac{{\rm Log}(z)}{{\rm Log}(u)}\ ]\ \} \tag{81}$$

$$J_3(v) \quad \equiv_{\rm df} \quad J_2(v)\ \wedge\ \forall u\ \exists z\ \{\ J_2(u)\ \supset\ [J_2(z)\ \wedge\ {\rm LogLog}(v) = \frac{{\rm LogLog}(z)}{{\rm LogLog}(u)}\ ]\ \} \tag{82}$$

The footnote[17] is intended only for those readers who are not previously acquainted with the prior literature about cut formulae (such as [13, 18, 24, 26, 28, 29, 38, 39]): it briefly reviews this literature by summarizing how α can formally prove the sentence $\Omega_{-1}^{J_0}$. We hope the reader will forgive us for not also reviewing how α has the capacity to similarly prove the sentences $\Omega_0^{J_1},\ \Omega_1^{J_2},\ \Omega_2^{J_3},$ .... . This inductive generalization is not discussed because we are attempting to keep the discussion in this appendix reasonably brief, and similar results have appeared often elsewhere in the prior literature [13, 18, 24, 26, 28, 29, 38, 39]. Roughly speaking, the intuitive reason that $\alpha \vdash \Omega_{i-1}^{J_i}$ is that α has an *ability to prove all* Peano Arithmetic's Π_1^- theorems: Its Π_1^- capacity makes α easier to analyze with more simple versions of cut formulae than would be possible under many of its more difficult counterparts in [13, 18, 24, 26, 28, 29, 38, 39] (Many of the axiom systems in

[17] As merely a simple illustrative example for the reader who may be unacquainted with the prior literature [13, 18, 24, 26, 28, 29, 38, 39], this footnote will show how to formally establish $\alpha \vdash \Omega_{-1}^{J_0}$. We first note that Equation (79) implies $\alpha \vdash J_0(x) \supset \forall p \exists q\ x = q - p$. Let us now use Solovay's trick and rewrite the preceding statement, with the names of its variables changed, so that it appears as: $\alpha \vdash J_0(x) \supset \forall q \exists r\ x = r - q$. Combining these two equations now yields: $\alpha \vdash J_0(x) \supset \forall p \exists r\ x = r - x - p$. Since α can prove all Peano Arithmetic's Π_1^- theorems, the preceding equation, combined with Equation (79), yields: $\alpha \vdash \forall x \forall y\ [\ (J_0(x) \wedge x = y - x)\ \supset\ J_0(y)\ ]$. Also Equation (79) trivially implies that $\alpha \vdash J_0(x) \supset \exists y\ x = y - x$. From the Hájek-Pudlák definitions of ω_0 and $\Omega_{-1}^{J_0}$, the last two equations imply $\alpha \vdash \Omega_{-1}^{J_0}$. Moreover, an easy inductive generalization of this paragraph will establish $\alpha \vdash \Omega_0^{J_1}$, $\alpha \vdash \Omega_1^{J_2}$, $\alpha \vdash \Omega_2^{J_3}$ ....

the cited articles do not prove all Peano Arithmetic's Π_1^- theorems, but yet they can still verify the existence of analogs of the sequence $J_0, J_1, J_2, ...$. See footnote [18] for an example.)

Theorem A.2 (below) will be proven by defining Derive(x, y) to be " $J_2(y) \wedge \text{HilbPrf}_\alpha(x, y)$ " and then applying Theorem A.1. Before providing more details, we should note that Solovay indicated that Pudlák's analysis of extensions [28] of Robinson's system Q should be credited for having stimulated Theorem A.2. The contribution of Wilkie-Paris [38]'s research will be evident from the manner in which the proof of Theorem A.2 employs their article. Nelson's work can strengthen Theorem A.2 significantly (see footnote 18).

Theorem A.2. (Solovay's Extension of Pudlák's Theorem): Suppose the syntax for arithmetically encoding proofs uses B-adic numbers. (Such encodings were used by Hájek, Nelson Paris, Pudlák and Wilkie [13, 24, 38]: It is essentially the same as the encoding of proofs as "byte-strings" in Appendix B.) Suppose α is a Successor-Regular axiom system. Then α cannot prove $\forall p \ \neg\text{HilbPrf}_\alpha (\ulcorner 0 = 1 \urcorner , p)$.

Proof. For the sake of establishing a proof-by-contradiction, let us temporarily assume that:

$$\alpha \vdash \quad \forall p \ \neg\text{HilbPrf}_\alpha (\ulcorner 0 = 1 \urcorner , p) \tag{83}$$

For any fixed constant i , define $\text{Der}_i(x)$ to be the formula $\exists y \ J_i(y) \wedge \text{HilbPrf}_\alpha(x, y)$, where J_i is some definable cut specified earlier in this section. Equation (83) then implies:

$$\alpha \vdash \quad \neg\text{Der}_i (\ulcorner 0 = 1 \urcorner) \tag{84}$$

Using any fixed $i \geq 2$, the footnote [19] explains why the cuts formulae $J_2, J_3, J_4...$ satisfy:

$$\alpha \vdash \Omega_1^{J_i} \quad \text{(for any fixed } i \geq 2 \text{)} \tag{85}$$

Readers familiar with the prior literature, such as the work of Wilkie-Paris [38], will note Equation (85) implies (see footnote[20]) that the pair $(\alpha , \text{Der}_2(x))$ meet Theorem A.1's three requirements.

[18] Nelson [24] has shown how an axiom system Q_0 , defined in Chapter 6 of his book, can prove analogs of $\Omega_{-1}^{J_0}$. $\Omega_0^{J_1}$, $\Omega_1^{J_2}$, $\Omega_2^{J_3}$, ... despite the fact that Q_0 is *so unusually weak* that it is unable to prove *even that* Addition and Multiplication satisfy the associative, commutative and distributive principles. The only price Nelson pays for establishing this fact is that his analogs of the cut formulae J_0, J_1, J_2, ... have more complicated definitions than our cut formulae in Equations (79)-(82). Thus, our preceding discussion could actually drop the assumption that α must prove all the Π_1^- theorems of Peano Arithmetic, if one were willing to employ more complicated variants of cut formulae, as Nelson [24] has already done.

[19] Our earlier discussion indicated $\alpha \vdash \Omega_{i-1}^{J_i}$. Since by a trivial argument $\alpha \vdash \Omega_{i-1}^{J_i} \supset \Omega_1^{J_i}$ when $i \geq 2$, Equation 85 follows.

[20] This is intuitively because Wilkie-Paris [38] established that if an axiom system has the power of $I\Sigma_0 + \Omega_1$, uses a Δ_0^- formula to recognize its axioms and employs a B-adic encoding scheme for coding its Hilbert-style proof, then it will satisfy the three Hilbert-Bernays-and-Löb-like conditions mentioned in the hypothesis of Theorem A.1. Moreover, if α is a Successor-Regular axiom system, this result also applies to extensions of $\alpha + \Omega_1$ (because such Successor-Regular systems α have a capacity to prove all Peano Arithmetic's Π_1^- theorems). Since Equation (85) indicates that that α can prove $\Omega_1^{J_i}$, one can trivially generalize the Wilkie-Paris formalism to verify that α can prove that $\text{Der}_i(x)$ satisfies these three Löb-like properties (for any fixed constant $i \geq 2$). In particular, $\text{Der}_2(x)$ satisfies the three Löb properties. (Since it should be evident to a reader familiar with the Wilkie-Paris formalism that α has these capacities, we omit further details so our summary of Solovay's proof can be kept reasonably short.)

The preceding paragraph, combined with Theorem A.1, enables us to complete Theorem A.2's proof-by-contradiction. This is because Theorem A.1 does not allow a Successor-Regular axiom system to have Der_2 satisfy the theorem's Conditions (1)-(3) and to *simultaneously prove* $\neg\mathrm{Der}_2(\ulcorner 0 = 1 \urcorner)$. Yet precisely these very conditions are established in the preceding paragraph. The forced conclusion from this contradiction is that the first sentence in the preceding paragraph was incorrect when it temporarily entertained the hypothesis that a consistent Successor-Regular axiom system α could satisfy Equation (83). □

Added Comments. Theorem A.2 can be strengthened considerably. Although the definition of a "Successor-Regular" axiom system requires that $\mathrm{Prf}_\alpha(x, y)$ to be a Δ_0 formula, this assumption can be dropped. It is also unnecessary for α to have a capacity for proving all the Π_1^- theorems of Peano Arithmetic: Instead, it needs to prove just some finite set of these theorems. It is also apparent that many of Pudlák's more general Incompleteness properties [28] for extensions of Robinson's system Q can be hybridized with Theorem A.2 (involving the Wilkie-Paris form of Restricted Herbrand proofs).

We omit these topics because Theorem A.2 is intended only to delineate those aspects of the broader research of Nelson, Paris, Pudlák, Solovay and Wilkie that are directly germane to ISREF(A) and IS(A). Theorem A.2 is relevant because all the seemingly natural hybrids of the ISREF(A) and IS(A) are "Successor-Regular" systems. Thus, Theorem A.2 establishes the quite counter-intuitive result, that any effort to hybridize these two systems will automatically fail because the resulting hybrid cannot be self-verifying.

Appendix B: The Turing-Function Encoding of Group-3 Axioms

This appendix will explain how to encode the Group-3 axioms for IS(A), $\mathrm{IS}^\lambda(A)$ and ISREF(A). Rogers [30] and Jeroslow [15] have already noted that Kleene's Fixed Point Theorem [16] can be used to expand any initial r.e. axiom system α into a broader system α^*, whose one additional axiom is

The union of the system α **WITH THIS ADDITIONAL SENTENCE** is a consistent axiom system.

The reason such systems α^* have not occurred very often in the prior literature is that they are typically inconsistent, despite the fact that they axiomatically do justify their own consistency (as was explained in Section 1's first paragraph). The most novel aspects of IS(A), ISREF(A) and $\mathrm{IS}^\lambda(A)$ will thus *not be* that they contain Group-3 axioms, similar to the indented sentence above. Rather, it is that they are actually consistent (as demonstrated by Theorems 3.4, 4.3 and 5.1).

The reader should briefly skim Section 4's second paragraph before reading this Appendix. (It formally defines our variant of a Semantic Tableaux proof.)

Our description of the encoding of the Group-3 axioms will be divided into two parts. This appendix will describe a variant IS(A), ISREF(A) and IS$^{\lambda}$(A), called their **Turing Versions**, whose Group-3 axiom schemas are especially easy to encode. The price for this simplicity is that the Turing variations of the Group-3 axioms will use a slightly broader set of atomic functions than those described in Section 2. Appendix C will illustrate a more mathematically sophisticated encoding of the Group-3 axioms, using Section 2's Grounding functions. This will be called the **Grounding Versions** of the Group-3 axioms.

In order to explain the difference between these two versions, it is desirable to first define a 4-tape universal Turing machine $\mathcal{M}$ that one of our axiom systems will simulate. The machine $\mathcal{M}$ will use "left-sided tapes", i.e. tapes that can be traversed infinitely far to the left, but not to the right of the address Zero.

The alphabet Σ stored on the Turing tapes will consist of three letters, denoted as "0", "1" and "2". The first two letters will describe the bit values of "0" and "1". They will be called the **Non-Null letters.** The letter "2" will be a null-value symbol. It will be assumed that only a finite number of letters on any tape are non-null symbols. The rightmost letter on the tapes will always be a null marker (to indicate that the tape head cannot move any further right). The remaining null values on the tape will appear as an infinite sequence of consecutive null markers on the tape's left side. The symbol " * " will denote this infinite repeating sequence of the letter "2". Thus, if $b_n, b_{n-1}, ...b_1$ represents a sequence of non-null bits then $*, b_n, b_{n-1}, ...b_1, 2$ indicates how this sequence will be stored on its Turing tape.

Define the sequence $b_n, b_{n-1}, ...b_1$ to be a **Normalized Representation** of the integer J iff its length $n = \text{Max}(\,1\,,\,1 + \lceil \text{Log}_2 J \rceil\,)$ and $J = \sum_{i=1}^{n} b_i \cdot 2^{i-1}$. The advantage of the preceding definition is that each natural number J can be mapped onto an unique normalized representation encoding it as a bit string (because its rightmost non-null bit b_n always equals 1). For simplicity, we assume in this section that all four tapes of the $\mathcal{M}$ initially contain such normalized bit strings.

One of our Turing tapes, denoted as $T_{\mathcal{E}}$, will store an "encoding" of the machine $\mathcal{M}$'s instruction set. The other tapes, T_1, T_2 and T_3, are input tapes, containing the data that should be processed. We will assume $\mathcal{M}$ is a *"Universal"* 4-tape Turing Machine, in that it satisfies the following criteria:

1. Given any 3-input partial recursive function $f(x_1, x_2, x_3)$, the machine $\mathcal{M}$ can simulate f by loading some instruction encoding e_f on the tape $T_{\mathcal{E}}$ and loading the triple (x_1, x_2, x_3) on M's input tapes. (The tape T_1 will store $f(x_1, x_2, x_3)$'s value at the time when $\mathcal{M}$ halts.)

2. Let the partial function $g(x_1, x_2, x_3)$ denote the running time on the random access machine $\mathcal{R}$ of the partial recursive function f . Then e_f's running time on $\mathcal{M}$ will be no worse than $k \cdot g(x_1, x_2, x_3)^k$, for some fixed constant k whose value depends only on $\mathcal{M}$ and $\mathcal{R}$.

Let Z denote an arbitrary 4-tuple of integers, which represent the transcribed numbers stored on $\mathcal{M}$'s four tapes at the time execution commences. Our Turing Machine simulation functions will include:

a. $Tape_i(Z, x, t)$ designating a number 0, 1, or 2, according to which letter is stored on $\mathcal{M}$'s i-th tape in position x at time $t-2$ when the 4-tuple Z designates the initial configuration on $\mathcal{M}$'s tapes.

b. $Head_i(Z, t)$ indicating the address of the i-th tape's head at time $t-2$ when the 4-tuple Z designates the initial configuration on $\mathcal{M}$'s tapes.

c. $State_i(Z, t) = 1$ if the machine $\mathcal{M}$ is in state i at time $t-2$ when the 4-tuple Z designates the initial configuration on $\mathcal{M}$'s tapes. $State_i(Z, t) = 0$ otherwise.

Only a finite number of Π_1 axioms are needed to define the Turing functions. For instance, the footnote [21] illustrates one possible encoding. The reason Items (a)-(c) refer to a time "$t-2$" rather than "t" is simply that the inequalities $\text{Tape}_i(Z, x, t) \leq 2$, $\text{State}_i(Z, t) \leq 1$ and $\text{Head}_i(Z, t) \leq t$ will then imply that the Turing functions satisfy Section 2's definition of a "Non-Growth Function".

Section 2 had indicated the particular "Grounding" functions, used by the Group-1 schema, were chosen in a fairly arbitrary manner. It also indicated that if $\mathcal{F}$ was any arbitrary set of non-growth functions defined by a finite set of Π_1 axioms, then it was permissible to add this set of functions of $\mathcal{F}$ to the Group-1 schema. Thus, one can add all the Turing functions to Section 2's Group-1 schema. Since the Turing functions are non-growth functions, all our prior theorems are guaranteed to remain valid.

Our remaining discussion will thus refer to two versions of IS(A), ISREF(A) and IS$^\lambda$(A), called their **"Turing"** and **"Grounding"** versions. The Group-1 axiom schema of the Grounding versions were defined in Section 2: Its function symbols represent only the Grounding functions. On the other hand, the Turing versions of these systems will recognize the function symbols for both Grounding and Turing functions.

Recall a formula was said to be Δ_0^- iff all its quantifiers are bounded and it was built with the use of the Grounding functions. Henceforth the same formula will be called Δ_0^+ when both Turing and Grounding functions appear in it. (The distinction between the definitions of Π_1^- and Π_1^+ is analogous.) The remainder of this section and Appendix C will establish that:

[21] There are several possible definitions of the Turing functions. One will have four parts. Its first part will define $\text{Tape}_i(Z, x, t) = \text{State}_i(Z, t) = \text{Head}_i(Z, t) = 0$ when $t = 0$ or 1. The second fragment will describe the initial values of the Head and State functions for $t = 2$. The third fragment will employ the Count and Logarithm functions to assure that the bit sequences stored on the tapes $T_\mathcal{E}$, T_1, T_2, and T_3 for $t = 2$ correspond to the four integers associated with the 4-tuple Z. (This is possible to encode as a Π_1 sentence essentially because the i-th bit of the integer x equals $\text{Count}(x, i) - \text{Count}(x, i-1)$.) For $t \geq 3$, the fourth fragment will inductively define the values of the Head, State, and Tape functions at time t in terms of the prior values at time $t-1$. More details about the exact formulation of the Group-1 axioms are unimportant because any formulation of these axioms as a finite set of Π_1 sentences is equally suitable for our purposes.

1. The Group-3 axioms of Turing versions of IS(A), ISREF(A) and IS$^{\lambda}$(A) can be encoded as Π_1^+ sentences (if one so desires).
2. The Group-3 axioms of Grounding versions of IS(A), ISREF(A) and IS$^{\lambda}$(A) can be encoded as Π_1^- sentences (if one so desires).
3. The Group-3 axioms of Turing versions of IS(A), ISREF(A) and IS$^{\lambda}$(A) can be encoded as Π_1^- sentences (if one so desires).

We will establish the Result (1) in this section. Appendix C will establish (2) and (3). Note that the Result (3) strictly supersedes (1). However, it is still desirable to discuss the topic (1) first because its encoding methodology is significantly simpler than those for (2) and (3). (In particular, the distinction between a Π_1^+ and a Π_1^- encoding will be so striking that it will take Appendix C approximately half-a-dozen pages to undertake a task analogous to the 5-sentence proof of Theorem B.1.)

Like all Gödel encodings, our description will begin with a summary of the syntax employed. Our Hilbert-style proofs will employ a language consisting of twenty fairly standard symbols. Semantic tableaux proofs will employ one extra symbol. These symbols are listed below:

1. The standard symbols $\wedge$, $\vee$, $\neg$, $\supset$, $\forall$ and $\exists$, plus two symbols for bounded $\forall$ and $\exists$ quantifiers.
2. The three types of left and right parenthesis symbols: (,), [,], {,and }.
3. The punctuation symbol " , " and the relation symbols " $=$ " and " $\leq$ ".
4. The symbols $\hat{C}$, $\hat{V}$, and $\hat{F}$ for designating a fixed constant, a variable and a function.
5. The symbol $\hat{U}$ for designating a "New Parameter" in a semantic tableaux proof. "New Parameters" are introduced into semantic tableaux proof trees during the elimination of an existentially quantified variable (see Section 4's second paragraph). This symbol is not used in Hilbert-style proofs.

Define a byte to be an unit consisting of six bits. Our twenty-one atomic symbols will be each assigned a separate 6-bit code, such that each will have their leftmost bit equal to "1". The constant representing the natural number i will be encoded as a string of $\lceil log_{32}(i+1) \rceil + 1$ bytes, where the first byte is the "$\hat{C}$" symbol and the remaining bytes encode i as a base-32 number, with the convention that the lead bit in each 6-bit sequence is "0". The i-th variable, i-th parameter, and i-th function symbols will be encoded with the same convention, except that their first byte will be the $\hat{V}$, $\hat{U}$ and $\hat{F}$ symbols, instead of $\hat{C}$. A function $\hat{F}_i$ with j input arguments will use an expression of the form [, , ,] to specify its inputs. Define *an integer encoding of a formula* as a number in base 64, representing its string of bytes.

A Hilbert-style proof will be defined to be a list of formulae, each of which is either an axiom or a deduction from its predecessor via Modus Ponens or Generalization [7, 13, 14, 23]. We will separate the formulae on this list by having each begin with a left-curly bracket symbol and terminate with a right-curly bracket symbol.

The curly bracket symbols will have a slightly different function for Semantic Tableaux proofs [10, 32] because these proofs have a tree-like rather than list-like structure. Each curly-left bracket will correspond to a node of the Semantic Tableaux's proof tree. We shall assume a particular node's stored sentence appears immediately to the right of its curly-left bracket symbol. If x and y designate two curly-left bracket symbols, we will view the node x as a descendent of y if the range associated with y's pair of curly brackets properly contains x's range. (This convention is merely the natural manner for representing a tree employing the parenthesis notation.) The main point is that the *integer encoding of either a Hilbert-style or Semantic-Tableaux-style proof* is defined as a number in base 64, representing its string of bytes.

We will also employ the notation defined below:

i. Subst (g , h) will denote Gödel's classic substitution formula, which yields TRUE when g is an encoding of a formula, and h is an encoding of a sentence which replaces all occurrence of free variables in g with simply a constant, whose value equals the integer g.

ii. UNION(A) will denote the union of the Group-Zero, Group-1 and Group-2 axioms.

iii. $\text{SubstSemPrf}_{\alpha}(s, y, g)$ is a formula stating that y represents a Semantic Tableaux Proof Tree of the theorem s from the union of the axiom system α with the one further axiom whose Gödel number is the integer h satisfying $\text{Subst}(g, h)$. The special case of SubstSemPrf where α corresponds to UNION(A) is denoted as $\text{SubstSemPrf}_{\text{UNION}_{(A)}}(s, y, g,)$.

Recall Section 2 indicated that IS(A) contained only one single axiom sentence in its Group-3 schema. This sentence was "informally" described as being equivalent to the statement:

$$\forall\, y \quad \neg\, \text{SemPrf}_{\text{IS(A)}}(\ \ulcorner\, 0 = 1 \,\urcorner\ ,\ y\ ) \tag{86}$$

The Group-3 axiom of IS(A) is easy to "formalize" because (86) is the only sentence belonging to IS(A)'s Group-3 schema. In particular, let $\Theta(g)$ denote the formula:

$$\forall y \quad \{\ \neg\ \text{SubstSemPrf}_{\text{UNION(A)}}\ (\ \ulcorner\, 0 = 1 \,\urcorner\ ,\ y\ ,\ g\ )\ \ \} \tag{87}$$

Let N denote the Gödel number of the formula $\Theta(g)$ above. Then the "formal" encoding of IS(A)'s Group-3 axiom (86) can be simply defined as being the sentence $\Theta(N)$.

The remainder of this section will have two goals. One will be to show $\text{SubstSemPrf}_{\text{UNION(A)}}(s, y, g)$ and its analogs have Δ_0^+ encodings. The second goal will be to describe the precise analog of $\Theta(N)$ needed to encode the more complex Group-3 axiom schemes for ISREF(A) and IS$^{\lambda}$(A). (These Group-3 schemes will be more complicated than IS(A)'s counterpart primarily because they consist of an infinite number of axiom-sentences, all referring to each other.) We will discuss the second topic first.

Define a **Pseudo-Formula** to be a logical construct which differs from conventional formula by allowing the three pseudo-symbols, denoted as " $\clubsuit$ ", " $\ulcorner \clubsuit \urcorner$ ", and " $\clubsuit_x^z$ ", to appear anywhere in the pseudo-formula. Usually, $\Xi(\clubsuit)$ will denote a pseudo-formula. The symbol " $\clubsuit$ " will be a place-holder where an arbitrary sentence Φ may be later inserted into the "pseudo-formula" Ξ. Similarly, the symbols " $\ulcorner \clubsuit \urcorner$ " and " $\clubsuit_x^z$ " will be place-holders for Φ's Gödel number $\ulcorner \Phi \urcorner$ and Section 2's symbol " Φ_x^z ". In particular, our discussion will include a formal encoding of the primitives:

iv. $\text{PseudoTransform}_1(h, r)$ indicating that h is a pseudo-formula, r is a conventional formula and there exists a Π_1^- sentence Φ such that each appearance of " $\ulcorner \clubsuit \urcorner$ " in h is replaced by an appearance of " $\ulcorner \Phi \urcorner$ " in r and each appearance of " $\clubsuit$ " is replaced by " Φ ".

v. $\text{PseudoTransform}_2(h, r)$ indicating that h is a pseudo-formula, r is a conventional formula and there exists a prenex* sentence Φ such that each appearance of " $\ulcorner \clubsuit \urcorner$ " in h is replaced by an appearance of " $\ulcorner \Phi \urcorner$ " in r and each appearance of " $\clubsuit_z^x$ " is replaced by " Φ_z^x ".

vi. $\text{ExSemPrf}_{\alpha}(s, y, g)$ is a generalization of item (iv)'s $\text{SubstSemPrf}_{\alpha}(s, y, g)$ formula, taking into account both pseudo-formulae and item (i)'s Subst relation. In particular, let h denote the unique Gödel number of a *pseudo-sentence* satisfying $\text{Subst}(g, h)$, where g is the Gödel number of a *pseudo-formula.* Then $\text{ExSemPrf}_{\alpha}(s, y, g)$ will indicate y is a Semantic Tableaux Proof of the prenex* sentence s, where each axiom a in the proof y satisfies either $a \in \alpha$ or $\text{PseudoTransform}_2(h, a)$.

vii. $\text{ExHilbPrf}_{\alpha}(s, y, g)$ will have a definition identical to $\text{ExSemPrf}_{\alpha}(s, y, g)$, except that it will treat y as a Hilbert-style proof, rather than as a Semantic Tableaux proof tree.

Added Comment. Theorems B.1 and Remark B.3 will imply these formulae have Δ_0^+ encodings.

Section 2 had indicated that $\text{IS}^{\lambda}(A)$ would contain one Group-3 axiom Γ_{Ψ} for each prenex* sentence Ψ. These axioms were previously "informally" described as:

$$\forall x \, \forall y \, \forall z \quad \{ \quad \text{SemPrf}_{\text{IS}^{\lambda}(A)} (\ulcorner \Psi \urcorner , \, y) \quad \wedge \quad y^{\lambda} < \frac{z}{x} \quad \supset \quad \Psi_z^x \quad \} \tag{88}$$

In order to "formally" encode (88), it is useful to employ the pseudo-formula $\Xi(\clubsuit)$ below. Note (89) differs from (88) by replacing "Ψ" with "$\clubsuit$"and replacing "$\text{SemPrf}_{\text{IS}^{\lambda}(A)}(\bullet, y)$" with "$\text{ExSemPrf}_{\text{UNION}(A)}(\bullet, y, g)$".

$$\forall x \, \forall y \, \forall z \quad \{ \quad \text{ExSemPrf}_{\text{UNION}(A)} (\ulcorner \clubsuit \urcorner , \, y , \, g) \quad \wedge \quad y^{\lambda} < \frac{z}{x} \quad \supset \quad \clubsuit_z^x \quad \} \tag{89}$$

Let N_{λ} denote the Gödel number of (89). Then the *informal expression* " $\text{SemPrf}_{\text{IS}^{\lambda}(A)} (s , y)$ " can be *formally encoded* as: " $\text{ExSemPrf}_{\text{UNION}(A)} (s , y , N_{\lambda})$ ". This implies that the *Formal Gödel Encoding*

of $\mathrm{IS}^{\lambda}(A)$'s Group-3 axioms (from (88)) is illustrated by Equation (90):

$$\forall\, x\, \forall\, y\, \forall\, z \quad \{ \quad \mathrm{ExSemPrf}_{\mathrm{UNION}_{(A)}} (\ulcorner \Psi \urcorner \, , \, y \, , \, N_{\lambda}) \quad \wedge \quad y^{\lambda} \leq \frac{z}{x} \quad \supset \quad \Psi_{z}^{x} \quad \} \tag{90}$$

Theorem B.1. The formula $\mathrm{ExSemPrf}_{\mathrm{UNION(A)}} (s \, , \, y \, , \, g)$ can be encoded as a Δ_0^+ formula.

Proof Sketch. We will not provide a formal proof of Theorem B.1 because it is fairly trivial and because the next section will provide a more detailed proof of a more powerful theorem (about Δ_0^- encodings).

The intuition behind Theorem B.1 can be appreciated when it is realized that every context-free grammar can be recognized in Polynomial time. Thus, a string with L symbols can be checked in Polynomial(L) time for whether it belongs to an arbitrary context-free grammar. In particular since y represents a string with approximately $\mathrm{Log}_{64} y$ bytes, one can determine in PolyLog(y) time whether or not y represents a tree whose individual nodes represent stored formulae using the language of $\mathrm{IS}^{\lambda}(A)$. Moreover, it is very trivial to modify this recognition procedure so that it can also check whether the triple (s, y, g) satisfies $\mathrm{ExSemPrf}_{\mathrm{UNION(A)}}(s, y, g)$) also in PolyLog($s + y + g$) time.

The **fundamental point is** $\mathrm{ExSemPrf}_{\mathrm{UNION(A)}}(s, y, g)$ must have a Δ_0^+ encoding *simply because* its Turing computation has a running time $\mathrm{PolyLog}(s + y + g) \ll \mathrm{O}(s + y + g)$. $\square$

Remark B.2. The significance of the Δ_0^+ encoding for $\mathrm{ExSemPrf}_{\mathrm{UNION(A)}} (s \, , \, y \, , \, g)$ is that it implies that $\mathrm{IS}^{\lambda}(A)$'s Group-3 axioms (in Equation (90)) can be encoded as Π_1^+ sentences. The proof of Theorem B.1 was kept abbreviated because Appendix C will offer an alternate proof of Theorem B.1's result, using a different set of atomic function symbols.

Remark B.3. A similar construction will provide a Π_1^+ encoding for ISREF(A)'s Group-3 axioms. In particular, let $\mathrm{ConSize}(x, y)$ denote a Δ_0^+ formula that is satisfied when y encodes a list of formulae and each constant in this list y represents a quantity strictly less than x. In this notation, the Group-3 schema for ISREF(A) can be *viewed informally as* axioms of the form:

$$\forall\, x\, \forall\, y \quad \{ \quad \mathrm{HilbPrf}_{\mathrm{ISREF}_{(A)}} (\ulcorner \Psi \urcorner \, , \, y \, , \, g) \wedge \mathrm{ConSize}(x, y) \quad \supset \quad \Psi_{x-1}^{x-1} \quad \} \tag{91}$$

Our formal Π_1^+ encoding for Equation (91) will once again begin with a pseudo-formula for representing it:

$$\forall\, x\, \forall\, y \quad \{ \quad \mathrm{ExHilbPrf}_{\mathrm{UNION}_{(A)}} (\ulcorner \clubsuit \urcorner \, , \, y \, , \, g) \wedge \mathrm{ConSize}(x, y) \quad \supset \quad \clubsuit_{x-1}^{x-1} \quad \} \tag{92}$$

Let N be (92)'s Gödel number. Then (93) gives the *formal encoding* of ISREF(A)'s Group-3 axiom for Ψ.

$$\forall\, x\, \forall\, y \quad \{ \quad \mathrm{ExHilbPrf}_{\mathrm{UNION}_{(A)}} (\ulcorner \Psi \urcorner \, , \, y \, , \, N) \wedge \mathrm{ConSize}(x, y) \quad \supset \quad \Psi_{x-1}^{x-1} \quad \} \tag{93}$$

The analog of Theorem B.1 implies that Equation (93) has a Π_1^+ encoding. Appendix C will show that more terse Π_1^- encodings are possible for each of IS(A), ISREF(A) and $\mathrm{IS}^{\lambda}(A)$.

Appendix C: The Π_1^- Encoding for the Group-3 Axioms

Our description of the Π_1^- encoding for the Group-3 axioms will be divided into three parts in this section. The definition of the linear computational hierarchy, known as " LinH, " will be reviewed. We will then prove that there exists computational procedures within this hierarchy that tests whether or not an arbitrary tuple (s, y, g) satisfies the conditions $\text{ExSemPrf}_{\text{UNION}(A)}(\, s\,,\, y\,,\, g\,)$ and $\text{ExHilbPrf}_{\text{UNION}(A)}(\, s\,,\, y\,,\, g\,)$. The last part of this section will generalize some prior theorems discussed by Hájek-Pudlák, Krajíček and Wrathall [13, 18, 46] to show that each decision procedure within LinH can be encoded by a Δ_0^- formula. (This will establish the existence of a Π_1^- encoding for IS(A)'s, ISREF(A)'s and IS$^\lambda$(A)'s Group-3 axioms.)

Some definitions of a non-deterministic Oracle Turing machine can be found in [13, 18, 46]. It is defined as a conventional non-deterministic multi-tape Turing machine, where one of the tapes is a specially designated Oracle tape, and the machine has three extra states, called s_Q, s_Y and s_N. Whenever the machine enters the state s_Q, it will check for whether or not the integer currently transcribed on the oracle tape is an encoding for a number lying in a specially prespecified language, called $\mathcal{L}$. The machine will subsequently enter the s_Y or s_N state, depending on whether the the answer to the preceding query is affirmative or not.

Throughout this section, we shall assume, that the tapes employed by our non-deterministic oracle machines are one-way tapes. These machines will be similar to Appendix B's multi-tape machines except that the machine $\mathcal{M}$ is now allowed to have a nondeterministic finite state control, to possess an oracle for an arbitrary prespecified language $\mathcal{L}$, and to contain any arbitrary prespecified finite number of tapes.

Let n denote the length of an input string x that a multi-tape Turing Machine will process. The following quite conventional definitions [13, 46] are employed:

1. **DTIME($f(n)$)** is the set of languages that some deterministic multi-tape Turing Machine can accept within the time bound $f(n)$.
2. **LinTime** is the union of DTIME($\,n\,$), DTIME($\,2\,n\,$), DTIME($\,3\,n\,$), ...
3. **NLinTime**($\,\sigma\,$) is similarly the set of languages that are accepted by a non-deterministic *linear time* Turing machine that has access to an oracle σ .
4. Σ_i^{lin} is defined to equal LinTime when $i = 0$, and it will equal NLinTime(Σ_{i-1}^{lin}) when $i > 0$.
5. **LinH** is defined to be the union of Σ_0^{lin} , Σ_1^{lin} , Σ_2^{lin} , ...

The preceding definition of the of the linear computational hierarchy, "LinH", is essentially the linear-time analog of the well-known Polynomial-Time Hierarchy.

Appendix B described our formal method for encoding a Semantic Tableaux Proof as a tree. It used a parenthesis notation to encode the tree where there was a 1-to-1 correspondence between the left-curly

bracket symbols " { " in our byte string and the nodes of the tree it represented. The $i-$th sentence of the proof-tree was required to appear just to the right of the corresponding " { " symbol.

Theorem C.1 There exists LinH algorithms to determine whether a byte-string (or in some instances whether a collection of several byte-strings) satisfies each of the twenty-five itemized conditions below. (The 25-th formula in this list is the " $\text{ExSemPrf}_{\text{UNION}_{(A)}}(s, y, h)$ " predicate that was employed in Equation (90)'s encoding of $\text{IS}^{\lambda}(A)$'s Group-3 schema.)

1. *Term(s)* indicating that the byte-string s is a term built out of Section 2's seven Grounding functions.
2. *Formula(s)* indicating s is a formula in the first-order language employing the Grounding functions.
3. *FormulaTree(t)* indicating that the byte-string s represents a tree where every node has one or two children and where the content s in each tree-node is a string satisfying Formula(s). (The formal encoding of the tree t will use the left and right curly bracket symbols, as was described earlier.)
4. *Sentence(s)* indicating the string s satisfies Formula(s) and has no free variables (i.e. it is a "sentence").
5. *SentenceTree(t)* indicating t satisfies FormulaTree(t) and its formulae s satisfy Sentence(s).
6. *Prenex*Sentence(s)* indicating that the byte-string s is a prenex* sentence.
7. Π_1-*Sentence(s)* indicating that the byte-string s is a Π_1^- sentence.
8. *Concatenate(x, y, z)* indicating the byte-string z is constructed by concatenating the strings x and y .
9. *Replace(s, t, x, y)* indicating that the byte-string s is identical to the byte-string t, *except that* every appearance of the substring x in s is replaced by an appearance of y in t.
10. *Replace$_k$* ($s, t, x_1, x_2, ...x_k, y_1, y_2, ...y_k$) designating the natural generalization of Replace(s, t, x, y), indicating that the byte-string s is identical to the byte-string t except that now k *distinct different substrings* $x_1, x_2, ...x_k$ from s have their every appearance in t replaced by the substrings $y_1, y_2, ...y_k$.
11. *Transform(a, b)* indicating that the byte-string a satisfies Formula(a) and that if Ψ designates this formula then b is the byte-string that represents the constant in IS(A)'s language that encodes $\ulcorner \Psi \urcorner$.
12. *Subst*(g , h) designating Gödel's classic substitution relation defined by Item (i) in Appendix B.
13. *Cardinality(t, n)* indicating t satisfies SentenceTree(t) and this tree is comprised of n nodes.
14. *IndexedSentence(i, s, t)* indicating that the byte-string t satisfies SentenceTree(t) and that the $i-$th node enumerated in this tree is the byte string s .
15. *Depth(i, d, t)* indicating t satisfies SentenceTree(t) and that the $i-$th node in this tree has a depth of d .
16. *Ancestor(i, j, t)* indicating t satisfies SentenceTree(t) and the $i-$th node in this tree is an ancestor of its $j-$th node.
17. *Sibling(i, j, t)* indicating that t satisfies SentenceTree(t) and its $i-$th and $j-$th nodes are siblings.
18. *Leaf(i, t)* indicating that the byte-string t satisfies SentenceTree(t) and its $i-$th node is a leaf.

19. *ProperRoot(s, t)* indicating that the byte-string t satisfies SentenceTree(t), that s encodes a prenex* sentence, and that the root of t is a second prenex* sentence identical to s except that the universal and existential quantifiers are reversed and one extra " $\neg$ " symbol appears to the right of the root's quantifiers. (In other words, t's root stores the sentence, negating s , as required by Section 4's definition of a Semantic Tableaux Proof for a prenex* normalized theorem s .)

20. *Closure(i, t)* indicating that the byte-string t satisfies SentenceTree(t) and that the subranch between t's root and $i-$th node contains a pair of contradictory sentences (i.e. some sentence " Υ " and its negation " $\neg\Upsilon$ ").

21. *Deduction(i, t)* indicating that the byte-string t satisfies SentenceTree(t) and its $i-$th node is a valid Semantic Tableaux Deduction from one of its ancestors using Section 4's deductive rules for $\wedge-$Elimination, $\neg-$Elimination, $\vee-$Elimination, $\supset-$Elimination, $\exists-$Elimination and $\forall-$Elimination.

22. *PseudoTransform*$_1(h, r)$ and *PseudoTransform*$_2(h, r)$ defined by Items (iv) and (v) in Appendix B.

23. *UnionAxiom(s)* indicating that the byte-string s is a Group-zero, Group-1 or Group-2 axiom.

24. *Group3Test*(s, g) indicating there exists some h satisfying both Subst(g, h) and PseudoTransform$_2(h, s)$. (We will employ Group3Test only when g corresponds to a special fixed constant $\bar{m}$ that is a template for generating Group-3 axioms. In this context, Group3Test$(s, \bar{m})$ will indicate s is a Group-3 axiom.)

25. *ExSemPrf*$_{\text{UNION}_{(A)}}(s, y, g)$ defined from the combination of the Items (ii) and (vi) of Appendix B. (Item (ii) defines "UNION(A)", and Item (vi) defines "ExSemPrf".)

Proof Sketch: Below are somewhat abbreviated descriptions of the twenty-five LinH procedures:

Description of LinH Decision Procedures for Representing Items (1) through (7): It is well known that context-free grammars can be recognized by LinH decision procedures (i.e. the needed LinH procedures require merely a non-deterministic Turing Machine with one tape and one extra stack[46]). Since Items (1) through (3) represent context-free grammars, these formulae have LinH representations.

The decision algorithm for Item (4) uses a 2-step procedure. It will first employ Item (3)'s procedure for checking that s satisfies Formula(s). If the preceding answer is affirmative then s will be accepted if a linear-time non-deterministic search is unable to find a variable in s which is free. (Such a "failed search" for an unquantified variable will indicate that s satisfies " Sentence(s) "). Since Item (4)'s procedure involves only performing one non-deterministic search after executing Item (3)'s procedure, it follows that Item (4)'s decision procedure lies at merely one level higher than Item (3)'s decision procedure in the LinH Hierarchy.

A similar argument where one moves one level higher up through the LinH Hierarchy will show that Items (5), (6) and (7) also have LinH procedures. □

Description of LinH Decision Procedures for Representing Items (8) through (12): It is trivial to construct LinH decision procedures for Items (8) through (11). Moreover, the LinH procedure for implementing Subst(g, h) is easy to construct by making subroutine calls to the procedures Transform, Formula and Replace (described in Items 4, 9 and 11). In the interests of brevity, we will omit giving a more detailed description for these Decision Procedures. □

Description of LinH Decision Procedures for Representing Items (13) through (20): Table II (located on the last page of this article) formally describes these eight LinH procedures. Each uses a similar 2-part LinH decision algorithm, where its first part checks to see if the string t satisfies SentenceTree(t). If the answer is affirmative, then the second step of these eight procedures will perform a second LinH search to determine whether some particular additional condition is also satisfied. See Table II for the formal details. □

Description of LinH Procedures for Deduction(i, t): Section 4 defined eight deduction rules for the " $\wedge$ ", " $\neg$ ", " $\vee$ ", " $\supset$ ", " $\exists$ " and " $\forall$ " symbols. Deduction(i, t)'s procedure will thus have eight parts to verify whether the $i-$th node is a permissible deduction under these rules. Each part will employ the predicates Term, Concatenate, Replace, IndexedSentence, Ancestor and Sibling in a fairly routine manner.

For instance, let " $\mathcal{A}$ " and " $\mathcal{B}$ " denote two special symbols, and let $\bar{m}$ denote a special fixed constant that formally represents the string " $\mathcal{A} \wedge \mathcal{B}$ ". The $\wedge-$Elimination Rule's Decision Procedure will accept the ordered pair (i, t) when a non-deterministic search can find four integers $k < i$, $q < t$ $r < t$ and $s < t$ satisfying Ancestor(k, i, t), IndexedSentence(k, r, t), IndexedSentence(i, s, t) and *either* Replace$_2(\bar{m}, r, \mathcal{A}, \mathcal{B}, s, q)$ *or* Replace$_2(\bar{m}, r, \mathcal{A}, \mathcal{B}, q, s)$.

The procedures for eliminating the symbols " $\neg$ ", " $\vee$ ", " $\supset$ ", " $\exists$ " and " $\forall$ " will be closely analogous to the LinH algorithm (above). Thus, the Elimination procedures for " $\vee$ ", and " $\supset$ ", will employ the Sibling LinH Decision Procedure in the natural manner suggested by these elimination rules. The $\forall-$Elimination Procedure will obviously employ the Term and Replace Subroutines to assure that a sentence of the form " $\forall v\, \Upsilon(v)$ " has all occurrences of " v " in " $\Upsilon(v)$ " replaced by *some term* " s " in the derived deduction " $\Upsilon(s)$ ". In the interests of brevity, we will omit the additional details. □

Description of LinH Procedures for PseudoTransform$_1(h, r)$ and PseudoTransform$_2(h, r)$: Both procedures have similar representations. Therefore we will describe only PseudoTransform$_1(h, r)$'s decision procedure. It will begin by calling the procedure Formula(r) to verify r is a formula. If the preceding answer is affirmative, then the ordered pair (h, r) will be accepted if a non-deterministic search can find an ordered pair of strings (a, b) with $a < r$ and $b < r$ satisfying each of Transform(a, b), Π_1-Sentence(a) and Replace$_2($ h , r , "$\clubsuit$" , "$\ulcorner \clubsuit \urcorner$" , a , b). □

Description of UnionAxiom(s)'s LinH Decision Procedure: For i equal to 0, 1 or 2, let Group$_i(s)$ be a formula indicating that s is a Group-i type axiom. It is trivial to devise a LinH algorithm that tests for whether Group$_0(s)$ is satisfied. Since there exists only a finite number of allowed Group-1 axioms, there certainly exists a LinH algorithm that can test for whether Group$_1(s)$ is satisfied. Employing PseudoTransform$_1$, it is easy to devise a LinH algorithm that also tests for whether Group$_2(s)$ is satisfied. (This is because for

some fixed constant $\bar{m}$, which serves as a "template" for "generating" the full set of Group-2 axioms, $\text{Group}_2(s)$ can be defined as the set of s satisfying $\text{PseudoTransform}_1(\bar{m}, s)$.) The decision procedure UnionAxiom(s) is then defined as the operation $\text{Group}_0(s) \vee \text{Group}_1(s) \vee \text{Group}_2(s)$. □

Description of Group3Test(s, g)'s LinH Decision Procedure: This procedure will accept the input (s, g) when a nondeterministic search, running in time $\text{O}(\text{Log}(s))$, can construct a string h such that both $\text{Subst}(g, h)$ and $\text{PseudoTransform}_2(h, s)$ are satisfied. (We may assume that our encoding scheme is designed so that $h \leq s$ when h exists. Therefore $\text{O}(\text{Log}(s))$ bounds the search's nondeterministic running time.) □

Description of $\text{ExSemPrf}_{\text{UNION}_{(A)}}(s, y, g)$'s LinH Decision Procedure: This decision procedure will begin by testing for whether SentenceTree(y) and ProperRoot(s, y) are satisfied. If these conditions are met, it will accept (s, y, g) if the two conditions below are satisfied

A. Every branch of the candidate-tree y is closed by containing a pair of contradictory sentences. (A LinH procedure for checking this condition will seek to non-deterministicly find an integer $i < y$ satisfying $\text{Leaf}(i, y) \wedge \neg \text{Closure}(i, y)$. The *inability to find such a* y will indicate that Condition (A) holds.)

B. Every non-root node in the candidate-tree y must be either an axiom of $\text{IS}^{\lambda}(A)$ or a permitted deduction from a higher node in the proof tree. A LinH procedure can verify this condition by confirming that a non-deterministic search *cannot* find two integers $i < y$ and $s < y$ satisfying:

$$i \geq 2 \ \wedge \ \text{IndexedSentence}(i, s, y) \ \wedge \ \neg \text{UnionAxiom}(s) \ \wedge \ \neg \text{Group3Test}(s, g) \ \wedge \ \neg \text{Deduction}(i, y) \quad (94)$$

Hence, $\text{ExSemPrf}_{\text{UNION}_{(A)}}(s, y, g)$ has a LinH Decision Procedure because the two conditions (above) can be tested by LinH procedures. □

Theorem C.2 For each decision procedure $D(\bullet)$ in the LinH Hierarchy, there exists a Δ_0^- formula $\Theta_D(s)$ such that $\Theta_D(s)$ is true precisely when the procedure D accepts the input string s .

Justification of Theorem C.2. It is implicit from the prior literature that Theorem C.2 is valid. Wrathall [46] showed how to translate LinH Decision Procedures into Rudimentary Formulae. Both Hájek-Pudlák and Krajíček [13, 18] discussed how Wrathall's result can be extended to establish that every LinH Decision Procedure has a Δ_0 encoding (using results from Benett's dissertation [2] as an intermediate step).

Our task is less ambitious and easier than the prior literature because it involves Δ_0^- rather than Δ_0 encodings. This is because COUNT(x,i) and LOG(x) are two of the seven Grounding functions allowed in Δ_0^- formulae. The axiom system $\text{I}\Sigma_0$ can also prove the COUNT operation is a total function represented by a Δ_0 graph, but the relevant proof in [13] is quite long. (As a mere intermediate step, the proof from

$I\Sigma_0$ in [13] requires employing some variant of Benett's encoding of Exponentiation [2], to verify that $I\Sigma_0$ can formally prove that Logarithm is a total function, represented by Δ_0 formula. This is necessary for [13] to define a function called "NUON(x)" satisfying $\text{NUON}(x) = \text{COUNT}(x, \infty)$.)

By choosing to be less ambitious and defining the Group-1 axioms so that they recognize two function symbols for Count and Logarithm, we simply avoid the hardest part of the proofs in [13, 18, 46]. The formal proof of Theorem C.2 appears in Appendix D. It is unnecessary for the reader to examine Appendix D, if he accepts the claim that Theorem C.2's proof is very implicit from the prior literature.

Theorem C.3. The Group-3 axiom schemas for $\text{IS}^{\lambda}(A)$, $\text{IS}(A)$ and $\text{ISREF}(A)$ have Π_1^- encodings.

Proof. The application of Theorem C.2 to Part 25 of Theorem C.1 implies $\text{ExSemPrf}_{\text{UNION}(A)}(s, y, g)$ has a Δ_0^- encoding. The remainder of the analysis of $\text{IS}^{\lambda}(A)$ is essentially identical to Appendix B's treatment of the same topic, except that we now do the proof in terms of Δ_0^- encodings rather than Δ_0^+ encodings. Thus, we again set N_λ equal to the Gödel number of the pseudo-formula in Equation (95), and define $\text{SemPrf}_{\text{IS}^{\lambda}(A)}(s, y)$ to be the formula : " $\text{ExSemPrf}_{\text{UNION}(A)}(s, y, N_\lambda)$".

$$\forall x \, \forall y \, \forall z \quad \{ \quad \text{ExSemPrf}_{\text{UNION}(A)} (\ulcorner \clubsuit \urcorner , \, y , \, g) \quad \wedge \quad y^{\lambda} < \frac{z}{x} \quad \supset \quad \clubsuit_z^x \quad \} \tag{95}$$

This definition, combined with Lemmas C.1 and C.2, immediately implies that " $\text{SemPrf}_{\text{IS}^{\lambda}(A)} (s , y)$ " has a Δ_0^- encoding, and that the corresponding Group-3 axioms for $\text{IS}^{\lambda}(A)$ are the Π_1^- sentences illustrated below:

$$\forall x \, \forall y \, \forall z \quad \{ \quad \text{ExSemPrf}_{\text{UNION}(A)} (\ulcorner \Psi \urcorner , \, y , \, N_\lambda) \quad \wedge \quad y^{\lambda} \leq \frac{z}{x} \quad \supset \quad \Psi_z^x \quad \} \tag{96}$$

Once again, the generalization of the preceding proof to $\text{IS}(A)$ and $\text{ISREF}(A)$ is routine. This is because analogs of Theorem C.1 imply that $\text{ExHilbPrf}_{\text{UNION}(A)}(s, y, g)$ and $\text{SubstSemPrf}_{\text{UNION}(A)}(s, y, g)$ also have LinH representations. Therefore applying Theorem C.2 to Equations (93) and (87), the Group-3 axioms of $\text{ISREF}(A)$ and $\text{IS}(A)$ have Π_1^- encodings similar to $\text{IS}^{\lambda}(A)$'s counterpart. □

Remark C.4. We want to return to the predicate $\text{Group3Test}(s, g)$ (defined in Part 24 of Theorem C.1) so that there is no ambiguity about its role in Theorem C.3's proof. Let the variables g and s correspond to the Gödel numbers of Equations (95) and (96) in Theorem C.3's proof. The latter Gödel number can be much larger than the former, and this issue may at first appear to be problematic in the context of an axiom systems that *recognizes no operation as a total function that grows faster* than Addition and Scalar Multiplication. However, such differences in magnitudes actually make no difference at all in the context of Theorem C.3's particular proof. This is because the proof defines N_λ to be the particular *fixed constant* representing Equation (95)'s Gödel number. In this context, Equation (96)'s formula " $\text{ExSemPrf}_{\text{UNION}(A)}(s, y, N_\lambda)$". will have *the constant* N_λ replace the variable g on every occasion where the subformula $\text{Group3Test}(s, g)$

appears inside this larger formula. Thus, the resulting formula " Group3Test (s , N_λ) " , has the desired property of being a Δ_0^- formula, free only in the variable s , representing the set of Gödel numbers corresponding to a sentence of the form of Equation (96).

Remark C.5 Recall that Section 2 had defined IS(A), ISREF(A) and IS$^\lambda$(A) so that their Group-1 function set could be possibly much broader than section 2's simple "Grounding Function Set". In particular, Group-1 could include the union of the Grounding functions with any further set of functions $\mathcal{F}$, where $\mathcal{F}$ may be any set of non-growth functions that can be defined by any finite set of Π_1 axioms. Theorem C.3 generalizes to such $\mathcal{F}-$extended versions of IS(A), ISREF(A) and IS$^\lambda$(A) in a perhaps surprisingly strong manner. This generalization will state that even when one adds the set of functions $\mathcal{F}$ to the Group-1 axioms, they are not needed to encode the Group-3 axioms as Π_1 sentences! For example, if $\mathcal{F}$ is Appendix B's set of Turing functions, then we can encode the Group-3 axiom schema as a set of Π_1^- sentences (rather than as a set of Π_1^+ sentences).

The preceding remark is surprising because it implies the Turing versions of the IS(A), ISREF(A) and IS$^\lambda$(A) have essentially no disadvantage. That is, these Turing versions clearly contain some added flexibility because their Group-1 axiom set is larger. One might initially suspect that a penalty for this additional flexibility would be that the Turing versions would require more cumbersome Π_1^+ Group-3 axioms. However, this is not the case. Our preceding construction can be generalized to establish that *even the Turing versions* of IS(A), ISREF(A) and IS$^\lambda$(A) can have their Group-3 axioms encoded as Π_1^- sentences!

Appendix D: Proof of Theorem C.2

Since close analogs of Theorem C.2 were previously described by Hájek-Pudlák, Krajícek and Wrathall [13, 18, 46], we will only sketch a proof of Theorem C.2 in this section.

Since the input string s has a bit-length equal to Logarithm(s), let us say that the decision procedure $D(\bullet)$ is a LinH procedure *With a Coefficient Strictly Bounded By* k when D requires no more than $k \cdot (\text{Log}_2(s) - 1)$ time to process any string s. Define a **ks-Vector** to be an ordered collection of k integers $u_1, u_2, ...u_k$, where each $u_i \leq s$. Henceforth, capital letter symbols, such as U, will denote ks-vectors. We will think of U as a sequence of $k \cdot (\text{Log}_2(s) - 1)$ bits, where its first $\text{Log}_2(s) - 1$ bits are stored in u_1, its second $\text{Log}_2(s) - 1$ bits are stored in u_2, ... etc. Thus the artificial notation " $\exists U$ " will be an abbreviated manner for formally writing $\exists u_1 \leq s, \exists u_2 \leq s, ... \exists u_k \leq s$. Similarly, " $\forall U$ " will be an abbreviation for $\forall u_1 \leq s, \forall u_2 \leq s, ... \forall u_k \leq s$. The advantage of the ks-vector notation is that it will describe how some particular memory bits (or other logical constructs) will change their state-values over the $k \cdot (\text{Log}_2(s) - 1)$ units of time during which the decision procedure D is running.

Let $U(t)$ denote the $t-$th bit of the ks-vector U. We will encode the action of our linear-time multi-tape Turing machine $\mathcal{D}$ by employing the following four groups of ks-vectors:

1. Assume that the Turing Machine $\mathcal{D}$'s finite state control has m bits of memory. For each $j \leq m$, there will exist a ks-vector B_j such that $B_j(t)$ indicates the value stored in the j-th bit of the finite state control at the time t.

2. Assume that each tape of the Turing Machine $\mathcal{D}$ consists of an infinitely long stream of bits. For each tape T_i, there will exist a ks-vector W_i such that $W_i(t)$ indicates which particular bit-value lies *underneath the tape's head* at the particular time t.

3. $L_i(t) = 1$ if the i-th Turing tape head moves left at the time t. Otherwise, $L_i(t) = 0$.

4. $R_i(t) = 1$ if the i-th Turing tape head moves right at the time t. Otherwise, $R_i(t) = 0$.

Let us assume that the relevant multi-tape Turing machines are using one-way tapes. Since the purpose of the machine $\mathcal{D}$ is to process the input string s, we will also assume that:

a) The tape T_1 initially stores a bit-string providing the binary encoding of the integer s.

b) The head of tape T_1 is initially located at the address $\text{Log}_2(s)$.

c) All the other tapes are initially filled with zero bits with their tape heads located initially at Address$= 0$

Recall that $\text{COUNT}(u, i)$ is one of the total functions recognized by the Group-1 axioms of $\text{IS}(A)$, $\text{ISREF}(A)$ and $\text{IS}^{\lambda}(A)$. It was defined to equal the number of "1" bits which are located to the right of the bit-address i in the integer u. Although $\text{COUNT}(u, i)$ is technically defined to be a function whose two input arguments are both required to be integers, we can trivially define a generalization of it, called $\text{COUNT}^k_s(U, i)$, which allows U to be a ks-vector. Our definition will require that k must be a prespecified constant whose value is known in advance and which is solely a function of $\mathcal{D}$. In this case (where k's value *is specified in advance),* the Group-1 axioms are sufficient for both encoding $\text{COUNT}^k_s(U, i)$ as Δ_0^- formula and for proving it is a total function. Below are a list of three useful features of the COUNT^k_s primitive:

I. For any ks-vector U, and for any $t \leq k \cdot (\text{Log}_2(s) - 1)$, we can use COUNT^k_s to formally calculate the value of the bit $U(t)$. This is because $U(t) = \text{COUNT}^k_s(t) - \text{COUNT}^k_s(t-1)$.

II. Let $\text{Head}_i(t)$ denote the address of the tape T_i's head at the time t. For any $t \leq k \cdot (\text{Log}_2(s) - 1)$, we can use COUNT^k_s to formally calculate the value of $\text{Head}_i(t)$. This is because

$$\text{Head}_i(t) \;=\; \text{COUNT}^k_s(L_i, t) \;-\; \text{COUNT}^k_s(R_i, t) \quad \text{when } i \geq 2 \tag{97}$$

and

$$\text{Head}_1(t) \;=\; \text{Log}(s) \;+\; \text{COUNT}^k_s(L_1, t) \;-\; \text{COUNT}^k_s(R_1, t). \tag{98}$$

III. Let $\text{Bit}_i(t, x)$ denote the bit stored at the address x of the tape T_i at the time t. For any $t \leq k \cdot (\text{Log}_2(s) - 1)$, we can use COUNT^k_s to formally calculate the value of $\text{Bit}_i(t, x)$. This is because Items (a) and (c) already indicated how to calculate the values for $\text{Bit}_i(0, x)$. For all other $t \leq k \cdot (\text{Log}_2(s) - 1)$, we can calculate the value of $\text{Bit}_i(t, x)$ by searching for the greatest $g \leq t$ such that $\text{Head}_i(g) = x$. If such a g exists then $\text{Bit}_i(t, x) = W_i(g)$. Otherwise, set $\text{Bit}_i(t, x) = \text{Bit}_i(0, x)$.

It is not hard to see how we can formally encode each of items I, II and III (above) as Δ_0^- formulae. In essence, whenever a ks-vector U appears in the context of an " $\exists U$ " or " $\forall U$ " quantifier, we can formally rewrite the quantifier as " $\exists u_1 \leq s$, $\exists u_2 \leq s$, ... $\exists u_k \leq s$ " or " $\forall u_1 \leq s$, $\forall u_2 \leq s$, ... $\forall u_k \leq s$ ".

Thus, the preceding discussion has outlined how we can encode any linear time decision procedure $D(s)$ of a Multi-Tape Deterministic Turing Machine as a Δ_0^- formula. An essentially identical argument can be used for every Σ_i^{lin} level of the Linear Time Hierarchy. This is because each level of the Linear Time Hierarchy may be represented by adding a finite number of additional " $\exists U$ " or " $\forall U$ " ks-vector quantifiers (all of which may be, once again, rewritten with bounded quantifiers).

We will not provide further details about the Δ_0^- encoding of LinH decision procedures in this Appendix because similar encodings have been discussed by Hájek-Pudlák, Krajícek and Wrathall [13, 18, 46]. It should be stressed that our COUNT(u,i) function is operationally equivalent to the "NUON" function [13], since $\text{NUON}(x) = \text{COUNT}(x, \infty)$. Therefore, the formal encoding, outlined in this appendix, has been very analogous to Chapter V.2.e of [13].

Table I: List of Π_1^- Axioms Defining The Grounding Functions and The Relation Predicates

1. $\forall x \;\; x = x$
2. $\forall x \; \forall y \;\; x = y \supset y = x$
3. $\forall x \; \forall y \; \forall z \;\; \{ \; x = y \wedge y = z \; \} \;\supset\; x = z$
4. $\forall x \; \forall y \quad x = y \quad \supset \quad \text{Predecessor}(x) = \text{Predecessor}(y) \quad \wedge \quad \text{Logarithm}(x) = \text{Logarithm}(y)$
5. $\forall x \; \forall y \; \forall a \; \forall b \quad \{ \; x = a \; \wedge \; y = b \; \} \quad \supset \quad x - y = a - b \quad \wedge \quad \frac{x}{y} = \frac{a}{b} \quad \wedge \quad \text{Count}(x, y) = \text{Count}(a, b) \quad \wedge$
   $\text{Maximum}(x, y) = \text{Maximum}(a, b) \quad \wedge \quad \text{Root}(x, y) = \text{Root}(a, b)$

6. $\forall x\ \forall y\ \forall a\ \forall b \quad \{\ x - y\ =\ a - b\ \ \wedge\ \ y = b\ \} \quad \supset \quad x = a$

7. $\forall x\ \forall y\ \forall a\ \forall b \quad \{\ x = a\ \wedge\ y = b\ \wedge x < y\ \} \quad \supset \quad a < b$

8. $\forall x\ \forall y \quad x = y\ \vee\ x < y\ \vee\ y < x$

9. $\forall x\ \forall y\ \forall z\ \ \{\ x < y \wedge y < z\ \}\ \supset\ x < z$

10. $\forall x \quad \neg\, x < x$

11. $\forall x \quad x < 1\ \ \supset\ \ x = 0$

12. Predecessor(0)=0

13. $\forall x \quad x \neq 0 \quad \supset \quad \text{Predecessor}(x)\ <\ x$

14. $\forall x\ \forall y\ \ \neg\ [\ \text{Predecessor}(x)\ <\ y\ <\ x\ ]$

15. $\forall x\ \forall y\ \exists z \leq y \quad x < y\ \supset\ x = \text{Predecessor}(z)$

16. $\forall x \quad x - 0 = x$

17. $\forall x\ \forall y\ \ y \neq 0\ \supset\ x - y\ = \text{Predecessor}(\ x\ -\ \text{Predecessor}(y)\ )$

18. $\forall x\ \forall y \quad x < y\ \supset\ \frac{x}{y}\ =\ 0$

19. $\forall x \quad \frac{x}{0}\ =\ \frac{x}{1}\ =\ x$

20. $\forall x\ \forall y \quad x \geq y \geq 1\ \supset\ [\ \frac{x}{y} > 0\ \wedge\ \frac{x}{y} - 1\ =\ \frac{x-y}{y}\ ]$

21. $\forall x\ \forall y\ \ \{\ x \geq y\ \supset \text{Maximum}(x, y) = x\ \}\ \wedge\ \{\ y \geq x\ \supset \text{Maximum}(x, y) = y\ \}$

22. $\text{Logarithm}(0){=}0 \wedge \text{Logarithm}(1){=}1 \wedge \text{Logarithm}(2){=}2$

23. $\forall x \quad x \geq 3\ \supset\ \text{Logarithm}(x) - 1\ =\ \text{Logarithm}(\ \frac{x}{2}\ )$

24. $\forall x \quad \text{Count}(x, 0)\ = \text{Count}(0, x)\ =\ 0$

25. $\forall x \quad x > 0 \quad \supset \quad \text{Count}(x - 1, 1)\ \neq \text{Count}(x, 1)\ \leq\ 1$

26. $\forall x\ \forall y \quad \{\ \text{Count}(x, 1) > 0 \vee \text{Count}(\frac{x}{2}, y - 1) > 0\ \} \quad \supset \quad \text{Count}(x, y) > 0$

27. $\forall x\ \forall y \quad \text{Count}(x, y) {-} \text{Count}(x, 1)\ = \text{Count}(\frac{x}{2}, y - 1)$

28. $\forall x \quad \text{Root}(0, x) = 0\ \wedge\ \ \text{Root}(x, 0)\ = \text{Root}(x, 1)\ =\ x$

29. $\forall x\ \forall y \geq 2 \quad \text{Root}(x, y)\ \leq\ \text{Root}(\ \frac{x}{\text{Root}(x,y)}\ ,\ y - 1\ )$

30. $\forall x\ \forall y \geq 2\ \forall z \quad z > \text{Root}(x, y) \quad \supset \quad z > \text{Root}(\frac{x}{z}, y - 1)$

Table II: Description of LinH Decision Procedures for Representing Items (13) through (20):

Cardinality(t, n)'s LinH procedure will count the number of " { " symbols appearing in the string t . It will accept the ordered pair (t, n) if this count$= n$ and SentenceTree(t) is satisfied.

IndexedSentence(i, s, t)'s LinH procedure will accept the string (i, s, t) if Sentence(s) and SentenceTree(t) are satisfied and the string lying between the $i-$th " { " symbol in the string t and the next curly bracket symbol in the tree t represents the string s .

Depth(i, d, t)'s LinH procedure will count the number of " } " symbols appearing to the left of the $i-$th " { " symbol in the string t . If k denotes this count then Depth(i, d, t)'s procedure will accept the triple (i, d, t) when $d = i - k - 1$ and SentenceTree(t) is satisfied.

Ancestor(i, j, t)'s LinH procedure will accept (i, j, t) if $i \leq j$ and a non-deterministic search can find two integers $d \leq j$ and $e \leq j$ satisfying Depth(i, d, t), Depth$(j, e, , t)$ and $d < e$, *but a second non-deterministic search cannot find* two additional integers $k < j$ and $f \leq d$ satisfying $k > i$ and Depth(k, f, t). (Also, the Ancestor procedure will accept the triple (i, j, t) if $i = j$ and Tree(t) is satisfied because any node will be considered as an "ancestor" of itself.)

Sibling(i, j, t)'s LinH procedure will accept (i, j, t) if a non-deterministic search can find $a < i$ and $d < t$ satisfying: Ancestor(a, i, t), Ancestor(a, j, t), Depth(i, d, t), Depth(j, d, t) and Depth$(a, d - 1, t)$.

Leaf(i, t)'s LinH procedure will make a non-deterministic search for some integer $b < t$ such that Ancestor(i, b, t) and $i \neq b$ are satisfied. It will accept the ordered pair (i, t) if the preceding LinH search fails but a non-deterministic search can find some $n < t$ satisfying Cardinality(t, n) and $i \leq n$.

ProperRoot(s, t)'s LinH procedure will accept (s, t) if Prenex*Sentence(s) and SentenceTree(t) are satisfied and the root of t represents a sentence r identical to s except that r's universal and existential quantifiers are reversed and one extra " $\neg$ " symbol appears directly to the right of the root's quantifiers.

Closure(i, t)'s LinH procedure will accept (i, t) if a nondeterministic search can find $a \leq i$, $b \leq i$, $s < t$ and $q < t$ such that Ancestor(a, i, t), Ancestor(b, i, t), IndexedSentence(a, s, t), IndexedSentence(b, q, t) and Concatenate(" $\neg$ ",q, s) are all satisfied (i.e. the ancestors of i represent some sentence " Υ " and its negation " $\neg\Upsilon$ ").

References

[1] Z. Adamowicz, "On Tableau consistency in weak theories", circulating manuscript from the Mathematics Institute of the Polish Academy of Sciences (1999).

[2] J. Benett, Ph. D. Dissertation, Princeton University, 1962.

[3] G. Boolos, The Logic of Provability Cambridge University Press, 1993.

[4] A. Bezboruah and J. Shepherdson, "Gödel's Second Incompleteness Theorem for Q", J of Symbolic Logic 41 (1976), 503-512.

[5] S. Buss, "Polynomial Hierarchy and Fragments of Bounded Arithmetic", 17th ACM Symp on Theory of Comp (1985) pp. 285-290

[6] S. Buss, Bounded Arithmetic, Princeton Ph. D. dissertation published in Proof Theory Lecture Notes, Vol. 3, published by Bibliopolic (1986).

[7] H. Enderton, A Mathematical Introduction to Logic, Academic Press 1972.

[8] S. Feferman, "Arithmetization of Metamathematics in a General Setting", Fund Math 49 (1960) pp. 35-92.

[9] S. Feferman,"Transfinite Recursive Progressions of Axiomatic Theories", J of Symbolic Logic, 27 (1962) 259-316.

[10] M. Fitting, First Order Logic and Automated Theorem Proving, Springer-Verlag Monograph in Comp Science, 1990.

[11] G. Gentzen, "Collected Papers of Gerhard Gentzen", (English translation by M. Szabo), North Holland (1969).

[12] P. Hájek, "On the Interpretatability of Theories Containing Arithmetic (II)", Comm. Math. Univ. Carol. 22 (1981) pp.595-594.

[13] P. Hájek and P. Pudlák, Metamathematics of First Order Arithmetic, Springer-Verlag (1991).

[14] D. Hilbert and B. Bernays, Grundlager der Mathematik Volume 1 (1934) and Volume 2 (1939), Springer.

[15] R. Jeroslow, "Consistency Statements in Formal Mathematics", Fundamentae Mathematicae 72 (1971) pp. 17-40.

[16] S. Kleene, "On the Notation of Ordinal Numbers", Journal of Symbolic Logic 3 (1938), pp. 150-156.

[17] J. Krajícek, "A Note on the Proofs of Falsehoods", Arch Math Logik 26 (1987) pp. 169-176.

[18] J. Krajícek, Bounded Propositional Logic and Complexity Theory, Cambridge University Press, 1995.

[19] J. Krajícek and P. Pudlák, "Propositional Proof Systems, The Consistency of First-Order Theories and The Complexity of Computation", J. of Symb Logic 54 (1989) PP. 1063-1079.

[20] G. Kreisel, "A Survey of Proof Theory, Part I" in Journal of Symbolic Logic 33 (1968) pp. 321-388 and " Part II" in Proceedings of Second Scandinavian Logic Symposium (1971) North Holland Press (with Fenstad ed.), Amsterdam

[21] G. Kreisel and G. Takeuti, "Formally self-referential propositions for cut-free classical analysis and related systems", Dissertations Mathematica 118, 1974 pp. 1 -55.

[22] M. Löb, A Solution to a Problem by Leon Henkin, Journal of Symbolic Logic 20 (1955) pp. 115-118.

[23] E. Mendelson, Introduction to Mathematical Logic, Wadsworth and Brooks/Cole Mathematics Series, 1987.

[24] E. Nelson, Predicative Arithmetic, Mathematical Notes, Princeton University Press, 1986.

[25] R.Parikh, "Existence and Feasibility in Arithmetic", Journal of Symbolic Logic 36 (1971), pp.494-508.

[26] J. Paris and C. Dimitracopoulos, "A Note on the Undefinability of Cuts", J of Symbolic Logic 48 (1983) pp. 564-569.

[27] J. Paris and A. Wilkie, "Δ_0 Sets and Induction", Proceedings of the Jadswin Logic Conference (Poland), Leeds University Press (1981) pp. 237-248.

[28] P. Pudlák, "Cuts Consistency Statements and Interpretations", Journal of Symbolic Logic 50 (1985) pp.423-442.

[29] P. Pudlák, "On the Lengths of Proofs of Consistency", in *Collegium Logicum: Annals of the Kurt Gödel Society Volume 2*, published (1996) by Springer-Wien-NewYork in cooperation with the Kurt Gödel Gesellshaft of the Institut für Computersprachen of Technische Universität Wien (Vienna Austria), pp 65-86.

[30] H. Rogers, Theory of Recursive Functions and Effective Compatibility, McGraw Hill 1967, see pp. 186-188.

[31] J. Shoenfeld, Mathematical Logic, Addison-Wesley, 1967.

[32] R. Smullyan, First Order Logic, Springer-Verlag, 1968.

[33] C. Smorynski, "The Incompleteness Theorem", in Handbook on Mathematical Logic, pp. 821-866, 1983.

[34] R. Solovay, Private Communications (April 1994) generalizing Pudlák's Proposition 2.2 from [28] to establish that no axiom system can recognize Successor, Subtraction and Division as functions, prove all the Π_1 theorems of Arithmetic and verify its own Hilbert consistency. (Solovay never published his theorem, and he gave us permission to present a short summary of his proof in Appendix A. Solovay's full theorem is slightly stronger than the result proven in Appendix A.)

[35] R. Statman, "Herbrand's theorem and Gentzen's Notion of a Direct Proof", in Handbook on Mathematical Logic, North Holland Publishing House (1983) pp. 897-913.

[36] G. Takeuti, "On a Generalized Logical Calculus", Japan Journal on Mathematics 23 (1953) pp. 39-96.

[37] G. Takeuti, Proof Theory, Studies in Logic Volume 81, North Holland, 1987.

[38] A. Wilkie and J. Paris, "On the Scheme of Induction for Bounded Arithmetic", Annals Pure Applied Logic (35) 1987, pp. 261-302.

[39] A. Wilkie, an unpublished manuscript which indicates that there exists a cut of Robinson's System Q that models $I\Sigma_0 + \Omega_n$ in a global rather than local manner. (The localized version of the same theorem was published by Nelson in [24].) Hájek and Pudlák [13] credit Wilkie's unpublished theorem (on page 407) for being responsible for Theorem 5.7 in Chapter V.5.c of their textbook, and they give a formal statement and proof of Wilkie's unpublished theorem.

[40] D. Willard, "Self-Verifying Axiom Systems", in Proceedings of the Third Kurt Gödel Symposium (1993) pp. 325-336, published in Springer-Verlag LNCS (Vol. 713). See also the next reference below:

[41] D. Willard, "Self-Verifying Axiom Systems and the Incompleteness Theorem", SUNY-Albany Technical Report March 1994. (This 50-page technical report expands the abbreviated proofs in the 12-page Extended Abstract [40] into full proofs.)

[42] D. Willard, "Self-Reflection Principles and NP-Hardness", Dimacs Series in Discrete Mathematics and Theoretical Computer Science (published by the American Mathematics Society), Volume 39 (December 1997), pp. 297-320.

[43] D. Willard, "The Tangibility Reflection Principle for Self-Verifying Axiom Systems", in The Proceedings of the Third Kurt Gödel Colloquium, (published in 1997 as Volume 1289 of Springer-Verlag LNCS Series), pp. 319-334.

[44] D. Willard, *The Semantic Tableaux Version of the Second Incompleteness Theorem Extends Almost to Robinson's Arithmetic Q*, presented initially as a 16–page conference abstract in Automated Reasoning with Semantic Tableaux and Related Methods (2000), Springer-Verlag LNAI#1847, pp. 415-430. A longer more detailed version of this paper will appear soon in the *JSL*. (It is approximately twice the size of the conference paper, and it contains some more detailed proofs and several further theorems.)

[45] D. Willard, *A Generalization of the Second Incompleteness Theorem and Some Exceptions to It*, submitted to the Bulletin of Symbolic Logic.

[46] C. Wrathall, "Rudimentary Predicates and Relative Computation", Siam J. on Computing 7 (1978), pp. 194-209.