ON THE AVAILABLE PARTIAL RESPECTS IN WHICH AN AXIOMATIZATION FOR REAL VALUED ARITHMETIC CAN RECOGNIZE ITS CONSISTENCY

DAN E. WILLARD

Abstract. Gödel’s Second Incompleteness Theorem states axiom systems of sufficient strength are unable to verify their own consistency. We will show that axiomatizations for a computer’s floating point arithmetic can recognize their cut-free consistency in a stronger respect than is feasible under integer arithmetics. This paper will include both new generalizations of the Second Incompleteness Theorem and techniques for evading it.


§1. Introduction. Let \( A(x,y,z) \) and \( M(x,y,z) \) denote two 3-way predicates indicating \( x + y = z \) and \( x \ast y = z \). An axiom system \( \alpha \) will be said to recognize successor, addition and multiplication as **Total Functions** iff it can prove:

\[
\forall x \exists z A(x, 1, z) \text{ AND } \forall x \forall y \exists z A(x, y, z) \text{ AND } \forall x \forall y \exists z M(x, y, z)
\]

Using this notation, we will say a formal system \( \alpha \) is of:

- **Type-S:** iff it contains an axiom declaring that successor is a total function
- **Type-A:** iff it contains an axiom declaring integer addition is a total function
- **Type-M:** iff it contains an axiom indicating integer multiplication is a total function.

This classification is related to Gödel’s Second Incompleteness Theorem. Thus, Solovay [??] noted how to combine some earlier methodologies of Nelson, Puldilák and Wilkie-Paris [??, ??, ??] to establish that all Type-S systems (that merely recognize addition and multiplication as 3-way relations) are unable to prove a theorem affirming their own Hilbert consistency, and Adamowicz-Zbierski [??, ??] developed a cut-free generalization of this result for \( I\Sigma_0 + \Omega_1 \). Subsequently, Willard [??] extended the semantic tableaux version of the Second Incompleteness Theorem to \( I\Sigma_0 \) and to virtually all Type-M systems. In contrast, [??, ??] showed exceptions to this tableaux version of the Second Incompleteness Theorem do exist for some Type-A systems.

The preceding research naturally raises the question whether an analogous phenomenon holds when one changes the venue of application from integer arithmetic to a computer’s commonly employed floating point arithmetic instruction set? In this paper, we will use the term **simulated real-arithmetic** to refer to an instruction set that is slightly more general and powerful than the common floating point instructions used by
a digital computer. We will prove that simulated real arithmetic is quite unlike integer arithmetic — insofar as an axiom system can simultaneously recognize its semantic tableaux consistency and the totality of all the simulated real-arithmetic operations.

Our result will be significant because a computer’s floating point instruction set has essentially as many practical applications as an integer arithmetic. Moreover, Section 3 will formalize another very unusual aspect of simulated arithmetic. It will be that our partial exceptions to the Second Incompleteness Theorem actually house a novel type of limited Gentzen-style deductive cut rule for simulated real arithmetic, which has no analog for integer-based arithmetics. Thus, the main contribution of this paper will be the demonstration that simulated real-valued arithmetic in several respects supports more robust forms of evasions of Gödel’s Second Incompleteness Theorem for cut-free deduction methods than an integer arithmetic can feasibly do.

It is, of course, obvious that if an axiomatization of simulated real arithmetic becomes sufficiently strong then the power of the Second Incompleteness Theorem will apply to it. However what is intriguing about simulated real arithmetic is that the threshold for activating the Second Incompleteness Theorem for cut-free deduction under simulated real arithmetic has a very different quality than its analog for integer arithmetic.

Thus, our analysis in this paper will consist of a dualistic description of both new threshold levels where the Second Incompleteness Theorem becomes applicable to simulated real arithmetic, as well as a description of when it can be evaded.

§2. General Formalism. The discussion in this abbreviated note will be largely self contained. Its gist should be comprehensible to a reader who has not yet read our earlier papers. Two 5-page literature surveys about the incompleteness properties of weak arithmetic systems have been provided in our earlier papers [??, ??], and a longer full length survey of this topic was written by Puldlák [??]. It is therefore unnecessary to include a lengthy survey about generalizations of the Second Incompleteness Theorem [??, ??, ??, ??, ??, ??, ??, ??, ??, ??, ??, ??, ??, ??] for weak arithmetics in this brief note. Instead, our discussion will cite the main results that we shall employ when they are used.

DEFINITION 1. Two formalizations of an integer, called NN and IPN, will be used in this paper. The first definition “NN” will represent the set of non-negative integers. The second definition, called “IPN”, will regard an integer as being any positive or negative whole number and also reserve a special symbol for representing $\infty$.

DEFINITION 2. The symbol $F$ will denote a 1-1 function that maps the set of NN integers onto IPN integers. In particular, let Even($x$) denote a function that equals 1 if $x$ is an even number and $-1$ if $x$ is odd. Let Half($x$) denote the integer-truncated quantity $\lfloor x \div 2 \rfloor$. Then $F(x)$ is defined by the convention that:

$F(x) = \text{Even}(x) \cdot \text{Half}(x)$ when $x \neq 1$ AND $F(1) = \infty$

---

1This result was first announced at the Tableaux-2005 Symposium (Springer-LNCS 3702 pp. 292–306).
Lower case letters $x$ will henceforth denote NN-integers, and upper case letters $X$ will denote IPN integers.

**Definition 3.** Let $i$ denote some indexing integer. Then the $i$–th **Simulated Real-Number** will be defined to be an ordered pair $(M_i, E_i)$ where $M_i$ is an IPN number storing the mantissa, and $E_i$ is a second IPN integer storing the exponent. The bold-face symbol $R_i$ will denote this simulated real-number. It is defined as follows:

1. If $E_i \neq \infty$ and $0 \neq M_i \neq \infty$ then $R_i = M_i \cdot 2^{-\lfloor \log_2(|M_i|) \rfloor} \cdot 2^{E_i}$.
2. If $E_i = \infty$ and $M_i$ is a power of 2, then $R_i$ represents the real number 0 written in a binary notation with $\log(M_i)$ digits to the right of the decimal point.
3. Otherwise, $R_i$ will represent an “overflow” (resulting from division by zero).

**Added Comment:** Often, the NN notation $(m_i, e_i)$ will be employed to denote a simulated real number, instead of IPN. In this case, Definition 2’s function $F$ will map $(m_i, e_i)$ onto its IPN counterpart $(M_i, E_i)$, so as to calculate $R_i$’s value.

**Definition 4.** Let $R_1$, $R_2$ and $R_3$ denote three simulated real-numbers that are encoded by the respective ordered pairs $(m_1, e_1)$, $(m_2, e_2)$ and $(m_3, e_3)$ when written in the NN-integer notation. Let $S$ denote one of the four arithmetic symbols of $+$, $\times$, $-$ or $\div$. Then $\Theta_S(m_1, e_1, m_2, e_2, m_3, e_3)$ will henceforth denote a formula which states that the two real numbers $R_1$ and $R_2$, combined under the arithmetic operation of $S$, will produce a third simulated real of $R_3$. More precisely, $\Theta_S(m_1, e_1, m_2, e_2, m_3, e_3)$’s definition will employ the usual computerized floating point hardware rounding convention that $R_3$’s computed mantissa has a bit-length $L$ equal to the maximum of the lengths for the two input mantissas of $R_1$ and $R_2$. It will thus specify $R_3$ represents the **closest approximation of the combination** of $R_1$ and $R_2$ under the operation $S$ which has a mantissa-length of $L$.

**Definition 5.** In a context where $R$ denotes the real number $(m_1, e_1)$, the term $\text{Expand}(R)$ will denote a second simulated real, $(m_2, e_2)$, whose value is identical to that of $R$ except that the mantissa for $\text{Expand}(R)$ will have one extra bit of precision (by containing an additional bit storing the value of zero). Thus, if $b_1$ denotes the rightmost bit of $m_1$ (when $m_1$ is viewed as an “NN” integer), then Equation (2) is a formula, denoted usually as $\Theta_E(m_1, e_1, m_2, e_2)$, indicating that $R_2 = \text{Expand}(R_1)$.

\[(2) \quad m_2 = 2 \cdot m_1 - b_1 \text{ AND } e_1 = e_2\]

The term **Simulated Real Arithmetic** will henceforth refer to an instruction set that includes the Expand operation (above) united with Definition 4’s four floating point operations for addition, subtraction, multiplication and division.

**Definition 6.** The predicate $\text{LongMult}(m_1, e_1, m_2, e_2, m_3, e_3)$ will denote that the real number $(m_3, e_3)$ represents the **untruncated** multiplicative product of the real numbers of $(m_1, e_1)$ and $(m_2, e_2)$. (Thus if $L^*$ equals the sum of the bit lengths for $m_1$ and $m_2$, then $m_3$’s bit-length will equal either $L^*$ or $L^* - 1$.) This is quite unlike Definition 4’s formalism which had $m_3$’s length be the maximum of $m_1$’s and $m_2$’s
lengths.) Thus unlike Definition 4’s formalism, Long Multiplication shall be viewed as lying formally outside the domain of the simulated arithmetic instruction set.

Finally, we will summarize the main results that will be presented in this article. Our first result was motivated by fact that [??] had established that no reasonable Type-M axiomatization of integer arithmetic can prove a theorem affirming its own semantic tableaux consistency. Despite this fact, we will show it is feasible to develop axiomatizations for simulated real arithmetic that can recognize their own cut-free consistency, as well as retain an ability to recognize the five arithmetic operations of simulated real arithmetic (given in Definitions 4 and 5) as total functions. Moreover, such systems will be able to additionally conceptualize integer addition, subtraction and division as total functions, as well as retain an ability to verify their consistency under a deductive method called Tab−1, which is a hybrid lying midway between semantic tableaux deduction and Hilbert deduction. Our second result will state that such partial exceptions for the Second Incompleteness Theorem do not also prevail under Hilbert deduction. (This is because no reasonable axiom system that merely recognizes Definition 4’s simulated real addition operation as a total function will be able to prove a theorem affirming its own Hilbert consistency.) Our final theorem will be a generalization of the Second Incompleteness Theorem. It will show that Definition 6’s notion of Long Multiplication differs from the truncated real multiplication (in Definition 4) by being incompatible with an axiom system recognizing its own semantic tableaux consistency.

§3. The Formal Nature of Simulated Arithmetic. It is useful to introduce some notation before discussing our new results. The operation \( F(a_1, a_2 \ldots a_j) \) will be called a **Non-Growth** function iff it satisfies: \( F(a_1, a_2 \ldots a_j) \leq \text{Maximum}(a_1, a_2 \ldots a_j) \). Six examples of non-growth functions are **Integer Subtraction** (where \( x - y \) is defined to equal zero when \( x \leq y \)), **Integer Division** (where \( x \div y \) is defined to equal \( x \) when \( y = 0 \), and it equals \( \lfloor x/y \rfloor \) otherwise), **Maximum**(\( x, y \)), **Logarithm**(\( x \)), **Root**(\( x, y \)) = \( \lceil x^{1/y} \rceil \) and **Count**(\( x, j \)) designating the number of “1” bits among \( x \)’s rightmost \( j \) bits. These function are called the **Grounding Functions**. The term **U-Grounding Function** will refer to a set of eight operations, which includes these six non-growth Grounding functions plus the **growth** operations of addition and \( \text{Double}(x) = x + x \). When discussing a formalism that employs NN integers and the U-grounding functions, our analogs for Logic’s \( \Pi_n \) and \( \Sigma_m \) sentences in the U-Grounding language will be called \( \Pi^*_n \) and \( \Sigma^*_m \) sentences. Under this notation, a **term** \( t \) is defined to be a constant, variable or a U-Grounding function symbol (whose input arguments are recursively defined terms). Also, the quantifiers in the wffs \( \forall v \leq t \Phi(v) \) and \( \exists v \leq t \Phi(v) \) are called **bounded integer quantifiers**. Any formula in the U-Grounding language, all of whose quantifiers are bounded, will be called \( \Delta^*_0 \). Following conventional notation, every \( \Delta^*_0 \) formula will be considered to satisfy the “\( \Pi_0 \)” and “\( \Sigma_0 \)” conditions. For \( n \geq 1 \), a formula \( \Upsilon \) shall be called \( \Pi^*_n \) iff it is written in the form \( \forall v_1 \forall v_2 \ldots \forall v_k \Phi \), where \( \Phi \) is \( \Sigma^*_n \). Likewise, \( \Upsilon \) is called \( \Sigma^*_n \).
iff it is written in the form \( \exists v_1 \exists v_2 \cdots \exists v_k \Phi \), where \( \Phi \) is \( \Pi_{n-1}^\alpha \). Also throughout our discussion, a sentence will be called “Tier(k)” when it is either \( \Pi_k^\alpha \) or \( \Sigma_k^\alpha \).

**Definition 7.** Let \( H \) denote a sequence of pairs \( (t_1, p_1), (t_2, p_2), \ldots (t_n, p_n) \), where \( p_i \) is a semantic tableau proof of the theorem \( t_i \), and let \( \mathcal{R} \) designate some class of sentences, such as perhaps Tier(k), \( \Pi_k^\alpha \) or \( \Sigma_k^\alpha \). In this context, \( H \) will be called a \( \text{Tab} \rightarrow \mathcal{R} \) proof of a theorem \( T \) from the axiom system \( \alpha \) iff \( T = t_n \) and also:

1. Each axiom in \( p_i \)’s proof is either one of \( t_1, t_2, \ldots t_{i-1} \) or comes from \( \alpha \).
2. Each of the “intermediate results” \( t_1, t_2, \ldots t_{n-1} \) lie in the pre-specified class \( \mathcal{R} \).

Thus, \( \text{Tab} \rightarrow \mathcal{R} \) deduction is stronger than classic semantic tableaux by allowing for a type of Gentzen-like deductive cut rule for sentences that belong to the intermediate class, that is formalized by \( \mathcal{R} \). Also, the symbol \( \text{Tab} \rightarrow k \) will denote the special version of \( \text{Tab} \rightarrow \mathcal{R} \) deduction that results when \( \mathcal{R} \) represents the Tier(k) class of sentences.

Using the preceding notation, it is possible to summarize [??, ??]’s main results. In essence, [??] showed that there are no reasonable axiom systems, employing the U-Grounding language, that can prove a theorem affirming their own self consistency under the Tab−2 deduction method. In contrast, [??] showed that some U-Grounding axiom systems can prove a theorem affirming their own consistency when Tab−1 deduction is used. In particular, [??]’s main result was the following assertion:

**Theorem 1.** Let \( A \) denote any axiom system employing the U-Grounding language whose theorems are valid under the standard model of the natural numbers. Then it is possible to construct a second “self justifying” and consistent axiom systems, called \( IS_D(A) \) in [??], with the following three properties:

1. \( IS_D(A) \) can prove all the Tier(1) theorems of \( A \).
2. \( IS_D(A) \) can prove all eight of the U-Grounding operations are total functions.
3. \( IS_D(A) \) contains an axiom affirming \( IS_D(A) \)'s consistency under Tab−1 deduction (i.e. that no Tab−1 formulated proof of \( 0=1 \) exists from \( IS_D(A) \)'s axiom-set).

The feature (3) of the axiom system \( IS_D(A) \) is unusual because Kleene [??] had observed that most axiom systems \( \alpha \) cannot be expanded into a broader system \( \alpha^* \), where an analog of Feature 3’s self-affirming axiom can declare \( \alpha^* \)'s own consistency. This is because such systems \( \alpha^* \) would be rendered inconsistent on account of essentially a Gödel-like diagonalization effect [??]. However, our article [??] explained that \( IS_D(A) \)'s self-affirming axiom is viable because \( IS_D(A) \) lies just below the threshold level where Gödel’s Second Incompleteness Theorem applies. We need two more lemmas to help prove our new results about simulated real arithmetic.

**Lemma 1.** For each of the four cases where \( S \) denotes one of Definition 4’s symbols of \(+, \times, \div \), the predicate \( \Theta_S(m_1, e_1, m_2, e_2, m_3, e_3) \) has a \( \Delta_0^\alpha \) encoding. Two other examples of predicates whose graphs have \( \Delta_0^\alpha \) encodings are:

1. Definition 5’s \( \Theta_E(m_1, e_1, m_2, e_2) \) predicate (for the “Expand” operator).
2. A predicate \( \Theta_S(m_1, e_1, m_2, e_2) \) indicating that the real number associated with \((m_1, e_1)\) is greater than or equal to \((m_2, e_2)\)'s real number.

Proof Sketch: It is helpful to begin our proof by distinguishing between the notions of \( \Delta_0 \), \( \Delta_0' \) and \( \Delta_0^* \) formulae. The first of these constructs has been studied extensively in the literature about arithmetic: It refers to the set of logical formulae whose function set includes only the addition and multiplication operators and all of whose quantifiers are bounded. The \( \Delta_0' \) class of formulae will differ from \( \Delta_0 \) by having no function operators. Instead, it will have four predicate symbols formalizing equality, less-than-or-equals, addition and multiplication. Thus, \( \Delta_0' \) will require that the upper limits associated with its bounded quantifiers be specified by a single variable rather than by a mathematical term. Finally, the second paragraph of this section had defined \( \Delta_0^* \).

Most of the prior research during the last 50 years has focused on the \( \Delta_0 \) class. An excellent summary of \( \Delta_0 \) encoding methods appears in the Chapter V of the Hájek-Pulduľák textbook [??] (including a summary of the techniques used by Bennett and Wrathall [??, ??]). Because \( \Delta_0 \) formulae are allowed to contain polynomial bounded quantifiers of the forms \( \forall v \leq x^k \) and \( \exists v \leq x^k \), one may apply routine methods to establish that \( \Theta_S \) has a \( \Delta_0 \) encoding in each of the six cases where \( S \) represents the symbols of \(+, \times, - , \div, E \) or \( G \). Moreover, Paris-Dimitracopoulos [??] showed how to translate every \( \Delta_0 \) formula into an equivalent \( \Delta_0^* \) representation. Since Equation (3) provides a \( \Delta_0^* \) definition for the graph for multiplication, it further follows that these \( \Delta_0^* \) predicates can be translated into equivalent \( \Delta_0^* \) forms.

\[
(3) \quad [(x = 0 \lor y = 0) \Rightarrow z = 0] \land [(x \neq 0 \land y \neq 0) \Rightarrow \left( \frac{z}{x} = y \land \frac{z - 1}{x} < y \right)]
\]

Thus, \( \Theta_S \) has a \( \Delta_0^* \) encoding in each of these six cases. \( \Box \).

Lemma 2. Let \( S \) again denote one of Definition 4's four arithmetic symbols of \(+, \times, - , \div \) under simulated real arithmetic. In each of these four cases, the statement that \( S \) represents a total function can be encoded as a \( \Pi_1 \) sentence. Also, there is a \( \Pi_1 \) sentence implying Definition 5's Expand operator is a total function.

Proof: It is clear that if Lemma 2’s statement was changed so that the totality of \( S \) was expressed as a \( \Pi_1 \) (rather than \( \Pi_1 \) ) sentence, then it would be a direct consequence of Lemma 1. This is because in each of the four cases where \( S \) denotes the symbol of \(+, \times, - , \div \), Equation (4) declares the totality of the operation of \( S \):

\[
\forall m_1 \forall e_1 \forall m_2 \forall e_2 \exists m_3 \exists e_3 \Theta_S(m_1, e_1, m_2, e_2, m_3, e_3)
\]

In order to construct a \( \Pi_1 \) sentence that implies the validity of (4), we will use the fact that in each of the four cases where \( S \) denotes the symbol of \(+, \times, - , \div \), a 6-tuple will satisfy \( \Theta_S(m_1, e_1, m_2, e_2, m_3, e_3) \) only when:

\[
** m_3 \leq \text{Double}(\text{Max}(m_1, m_2)) \land e_3 \leq \text{Double}(\text{Double}(\text{Max}(e_1, e_2)))
\]
Let $t$ denote the term of $\text{Double}(\text{Double}(\text{Max}(m_1, m_2, e_1, e_2)))$. Item ** then implies that Equation (5) is a $\Pi_1^0$ sentence which implies the validity of Equation (4).

(5) \[ \forall m_1 \forall e_1 \forall m_2 \forall e_2 \exists m_3 \leq t \exists e_3 \leq t \Theta_S(m_1, e_1, m_2, e_2, m_3, e_3) \]

Similarly, Equation (6) is a $\Pi_1^0$ formula implying $\text{Expand}(R)$ is a total function. \[ \square \]

**Theorem 2.** Let $A$ denote any axiom system employing the U-Grounding functions that is valid under the standard model of the natural numbers. Then there exists a second axiom system $A' \supseteq A$ such that Theorem 1’s axiom system $IS_D(A')$ is a consistent system, capable of verifying its own Tab-1 consistency and able to prove all five of Lemma 2’s simulated real arithmetic functions are total functions.

**Proof:** Let $\Psi_1, \Psi_2, \Psi_3, \Psi_4$ and $\Psi_5$ denote the five $\Pi_1^0$ sentences, defined by Lemma 2, that indicate the totality of the five simulated arithmetic functions. Let $A'$ denote the union of the axiom system $A$ with these sentences. Theorem 2 then follows immediately from the combination of Lemma 2 and Theorem 1. \[ \square \]

Some notation is needed to explain Theorem 2’s significance. Let $\langle m, e \rangle$ denote the simulated real with mantissa $m$ and exponent $e$. Also $| \langle m, e \rangle |^J$ will denote the quantity begotten by taking $\langle m, e \rangle$’s absolute value and raising it to the $J$–th power. Assuming that $J \neq 0$, $\langle n, f \rangle \geq 1$ and that $\text{Length}(m)$ denotes $m$’s bit-length, the formal expression of $\langle m, e \rangle \ll_L^J \langle n, f \rangle$ will have the following meaning:

1. $\text{Length}(m) \leq \text{Length}(n) + L$ and if $J \geq 1$ then $| \langle n, f \rangle |^J \geq | \langle m, e \rangle | \geq 1$.
2. $\text{Length}(m) \leq \text{Length}(n) + L$ and if $J \leq -1$ then $| \langle n, f \rangle |^J \leq | \langle m, e \rangle | \leq 1$.

Assuming that $R$ is a term that specifies the value of a simulated real number whose value is greater than 1, we will also use the preceding notation to define **Bounded Real Quantifiers** of the form $\exists \langle m, e \rangle \ll_L^J R$ and $\forall \langle m, e \rangle \ll_L^J R$.

The term **Bounded Integer Quantifier** will refer to expressions of the form $\forall x \leq t$ or $\exists x \leq t$, that were defined at the beginning of this section. A wff will be called $\Delta_0^\ominus$ iff it is built in any arbitrary manner out of our four forms of bounded quantifiers, together with the usual U-Grounding function symbols, the equality and less-than predicates and the standard Boolean connectives. If $\Psi$ is a $\Delta_0^\ominus$ formula then the expressions of $\forall v_1 \forall v_2 \ldots \forall v_k \Psi$ and $\exists v_1 \exists v_2 \ldots \exists v_k \Psi$ will be called $\Pi_1^\ominus$ and $\Sigma_1^\ominus$ formulae. Also a sentence will be called $\text{Tier}(1)^{\oplus}$ when it is either $\Pi_1^\ominus$ or $\Sigma_1^\ominus$.

**Theorem 3.** There exists a function $F$ that maps each $\text{Tier}(1)^{\oplus}$ formula $\phi$ onto a $\text{Tier}(1)$ formula $\Phi$ such that $\phi$ and $\Phi$ are equivalent to each other (under any axiom system that uses $\Phi$’s and $\phi$’s notation and which proves all $\Sigma_0$’s $\Pi_1^\ominus$ theorems).

**Proof Sketch:** Let $\Theta^L_1(m_1, e_1, m_2, e_2)$ be a predicate that is satisfied when the condition $\langle m_1, e_1 \rangle \ll_L^J \langle m_2, e_2 \rangle$ holds. Lemma 1 implies that $\Theta^L_1$ has a $\Delta_0^*$ encoding. In a context where $L' = L + 2$, $J' = \lceil \log(|J|) \rceil + 2$ and $\text{Double}^K$ denotes $K$ iterations of the Double function, it is easy to see $\langle m_1, e_1 \rangle$ can satisfy
∀ \langle \rangle of the type some fixed constant \( K \) that is bounded by an excessively tight constraint of the form \( K \cdot (1) \). 

This fact, in turn, implies that the following two invariants must hold:

A) The wff \( \exists \langle m_1, e_1 \rangle \iff \langle m_2, e_2 \rangle \Psi(m_1, e_1) \) is equivalent to the formula:
\[ \exists m_1 \leq \text{Double}^L(m_2) \exists e_1 \leq \text{Double}^J(e_2) [ \Theta^L(m_1, e_1, m_2, e_2) \land \Psi(m_1, e_1) ] \]

B) The wff \( \forall \langle m_1, e_1 \rangle \iff \langle m_2, e_2 \rangle \Psi(m_1, e_1) \) is equivalent to the formula:
\[ \forall m_1 \leq \text{Double}^L(m_2) \forall e_1 \leq \text{Double}^J(e_2) [ \Theta^L(m_1, e_1, m_2, e_2) \rightarrow \Psi(m_1, e_1) ] \]

Hence if we repeatedly apply the rules (A) and (B) to translate each bounded real quantifier into their equivalent forms, then at the end of this process we will have translated a Tier(1)\(^{2}\) formula \( \phi \) into an equivalent Tier(1) formula \( \Phi \). □

**Remark 1.** Theorem 3’s proof was of course quite simple. However, its implications for numerical analysis are actually quite subtle. The purpose of numerical analysis is essentially to produce Cauchy sequences of real numbers \( \mathbf{R}_1, \mathbf{R}_2, \mathbf{R}_3, \ldots \) that converge upon target answers with a decreasing error rate \( \epsilon_1, \epsilon_2, \epsilon_3, \ldots \). By choosing to use mantissas with sufficiently large lengths, it is easy to formalize such algorithms under simulated real arithmetic. Moreover, most of the theorems specifying the efficiency of numerical algorithms can be encoded as Tier(1)\(^{2}\) sentences. Thus, one can formally prove that if one sets the base \( A \) of the axiom system \( \text{IS}_D(A) \) equal to Peano Arithmetic, then \( \text{IS}_D(A) \) can prove all the Tier(1)\(^{2}\) numerical theorems that Peano Arithmetic can verify. Moreover since by Theorem 3, the Tab–1 deduction method can apply a modus ponens deductive rule to any Tier(1)\(^{2}\) sentence, it follows that \( \text{IS}_D(A) \) will retain a surprising ability to recognize the consistency of any chain of Tier(1)\(^{2}\) numerical theorems that are derived via this Tab–1 deductive cut rule !!!

**Remark 2.** It is useful to explore the difference between a purist branch of mathematics, such as Number Theory, and a highly computation-oriented subject, such as Numerical Analysis. Very few of the theorems about Number Theory can be encoded as Tier(1) sentences. This is because the Tier(1) bounded quantifiers, such as \( \forall x \leq t(v_1, v_2 \ldots v_j) \) or \( \exists x \leq t(v_1, v_2 \ldots v_j) \), will require that \( x \) span over a range that is bounded by an excessively tight constraint of the form \( K \cdot \text{Max}(v_1, v_2 \ldots v_j) \), for some fixed constant \( K \). In contrast, the bounded real quantifiers in a Tier(1)\(^{2}\) formula of the type \( \exists \langle m, e \rangle \iff \langle R \rangle \) or \( \forall \langle m, e \rangle \iff \langle R \rangle \) will allow \( \langle m, e \rangle \)'s real number to attain values essentially as large as \( \mathbf{R}^J \). This distinction is important because many of the classic results in Numerical Analysis, when reduced to the stage of producing Cauchy sequences that converge (with a specified efficiency) upon target real numbers, can be encoded as Tier(1)\(^{2}\) formulae on account of the polynomial range of the latter’s real-valued bounded quantifiers [??, ??]. Hence what we are suggesting here is that while a formalism, such as \( \text{IS}_D(A) \), will serve very few of the needs of a purist branch of mathematics, such as Number Theory, it can treat the computation-oriented Numerical Analysis more adroitly. Thus while Gödel’s Second Incompleteness Theorem certainly applies to both Number Theory and Numerical Analysis, it allows the
latter subject a greater degree of freedom to, under some circumstances, partially evade its reach.

§4. Three Related Incompleteness Results. Solovay [??] observed how one could modify an incompleteness theorem from Puldák’s article [??], with the techniques of Nelson and Wilkie-Paris [??, ??], to establish the following more refined result:

THEOREM 4. (Solovay’s 1994 refinement [??] of Pudlák’s Theorem 2.3 from [??])
No consistent Type-S axiom system that formalizes integer addition and multiplication as 3-way relations \( A(x,y,z) \) and \( M(x,y,z) \) satisfying the associative, distributive and idempotent axioms, can prove the non-existence of a Hilbert-proof of \( 0 = 1 \) from itself.

A 4-page summary of the idea [??] which Solovay privately communicated to us can be found in Appendix A of [??]. Although Theorem 4’s statement technically makes no mention of functions for simulating real-valued arithmetic, it is relevant to this topic because it delineates an important direction in which the formalisms of Theorems 1–3 cannot be further extended.

Henceforth, the term \( \Pi_{1}^- \) will refer to the subset of \( \Pi_{1}^+ \) sentences that do not contain any appearances of the Addition or Double function symbols. Thus, a formalized axiom system containing solely \( \Pi_{1}^- \) sentences will lie outside the gendre of Theorem 4’s formalism (because it does not include the axiom that successor is a total function). It is for this reason that Theorems 5 and 6 are of interest.

THEOREM 5. There exists a \( \Pi_{1}^- \) sentence \( W \) (with a very simple structure) such that no consistent Grounding-language based axiom system \( \alpha \), containing \( W \) as an axiom, is able to simultaneously prove a theorem affirming its own Hilbert consistency and to also prove that Definition 4’s simulated-real addition operation is a total function.

Proof Sketch: We will use Theorem 4 to justify Theorem 5’s validity. The key point is that the totality of the simulated addition operation implies that the operation that maps the simulated real \( R \) onto \( R + R \) is a total function. Note that the exponent of \( R + R \) exceeds the exponent of \( R \) by an increment of 1 (when these exponents are viewed as IPN integers). Assuming the \( \Pi_{1}^- \) sentence \( W \) is rich enough, the axiom system \( \alpha \) will be capable of deducing successor is a total function among NN integers from its totality among IPN integers. Also assuming \( W \) is rich enough, it will contain sufficient information to indicate that addition and multiplication among NN integers satisfies the associative, distributive and idempotent axioms mentioned in Theorem 4’s hypothesis. Thus under these circumstances, we can apply Theorem 4 to conclude that \( \alpha \) is unable to prove a theorem affirming its own Hilbert consistency.

THEOREM 6. There exists some particular \( \Pi_{1}^- \) sentence \( W \) such that no consistent axiom system \( \alpha \supset W \) will include the statement that Definition 6’s Long Multiplication operation is a total function among simulated real numbers and will also prove a theorem affirming the non-existence of a semantic tableaux proof of \( 0=1 \) from itself.
**Proof Sketch:** Our justification of Theorem 6 will use the fact that the article [?] had established the existence of a $\Pi^-_1$ sentence $V$ such that no consistent axiom system $\alpha \supset V$ will include the axiom that integer-multiplication is a total function and also prove a theorem formally affirming the non-existence of a semantic tableaux proof of $0=1$ from itself. The discussion in [?] technically focused on the topic of multiplication among NN integers. However, it is trivial to extend [?]’s results also to IPN integer-multiplication, provided one replaces [?]’s base starting axiom $V$ with a slightly revised $\Pi^-_1$ axiom $W$. Thus, our proof of Theorem 6 will be finished by employing the observation that two input real numbers $R_1$ and $R_2$, whose multiplicative product is $R_3$ (under Long Multiplication), will have the property that their mantissas (under IPN multiplication) will produce $R_3$’s mantissa. 

**Remark 3.** It is useful to compare the contrasting features of Theorems 2 and 6. Consider a sequence of real numbers $R_0$, $R_1$, $R_2$, ... defined by the recurrence rules that $R_0 = 2.0$ (with $k$ bits of precision) and that $R_{i+1} = R_i \ast R_i$. Then, $R_n$ represents the quantity $2^{n^2}$ stored with respectively $k + n$ and $O(k \cdot 2^n)$ bits of memory under Definition 4 and 6’s formalizations of multiplication. Thus if $n = 100$, then the first truncated representation of $R_n$ is tractable (since it uses $100 + k$ digits) — whereas the latter untruncated form of multiplication requires more digits than there are atoms in the universe. This example and its many analogs illustrate why a *computation-oriented subject*, such as Numerical Analysis, is forced to use Definition 4 rather than 6’s form of multiplication. Thus while Theorem 6’s generalization of the Second Incompleteness Theorem is surely significant, $R_n$’s example, combined with the other computational issues raised by Remarks 1 and 2, do illustrate that Theorem 2’s evasion of the Second Incompleteness Theorem also has some interesting features.

§5. **Overall Perspective.** Summing together all these results, we have illustrated how the threshold levels where the Second Incompleteness Theorem becomes active has a different quality for the two mathematical fields of Numerical Analysis and Number Theory. Thus, [?] has shown that Type-M axiomatizations for integer arithmetic are unable to recognize their semantic tableaux consistency — whereas Theorem 2 established that a *computation-oriented* real-valued axiom system can simultaneously recognize its Tab–1 consistency and the totality of the simulated arithmetic instruction set. Another contrast is that Remarks 1 and 2 showed that a goodly number of the theorems of numerical analysis fall into the Tier(1) and Tier(1)$^\oplus$ classes — while very few of the theorems of Number Theory fall into such categories. This fact is important because these two remarks also noted how the main robustness of an evasion of the Second Incompleteness Theorem will dwell within these two classes.

Naturally when an axiomatization for Numerical Analysis does become sufficiently strong, the Second Incompleteness Theorem will apply to it. However, the precise threshold where the Second Incompleteness Theorem becomes active has a different quality for integer arithmetic than for simulated arithmetic.
It is plainly obvious that conventional axiom systems possess many virtues that simulated arithmetics simply lack. For instance, the deficiency of an axiomatic framework that does not recognize integer-multiplication as a total function is quite readily evident. However, the salient point is that systems using simulated arithmetic also possess some partial (albeit very mixed) virtues. They are able to recognize their own Tab–1 consistency and to simultaneously focus on the computationally tractable components of numerical analysis.

In closing, the combination of our contrasting positive and negative results — where we have generalized Gödel’s Second Incompleteness Theorem and also identified some of its boundary-case exceptions — should help to sharpen the academic community’s knowledge about the meaning of Gödel’s historic discovery, by identifying the precise circumstances where Gödel’s centennial theorem is applicable.

Acknowledgment: I thank the anonymous referee for his suggestions.