GÖDEL'S INCOMPLETENESS THEOREM VIA UNDECIDABILITY OF THE HALTING PROBLEM

JOEL FRIEDMAN

Contents

Disclaimer: The material may sketchy and/or contain errors, which I will elaborate upon and/or correct in class. For those not in CPSC 421/501: use this material at your own risk...

1. The Point of this Article: Gödel's Incompleteness Theorem

Sipser's textbook [Sip] explains the rough idea behind a proof of a weaker form of Gödel's Incompleteness Theorem: namely, there exists a true but "unprovable" statement in $\mathrm{Th}(\mathbb{N}, +, \times)$. Here a "statement" (also "sentence") means a formula (meaning a "well-formed formula")[1] taking values in $\{T, F\}$ (T for true, F for false), that is TQ ("totally quantified"), i.e., where all variables are quantified[2].

Date: Tuesday 21st October, 2025, at 10:19.
Research supported in part by an NSERC grant.

[1]The notion of a WFF (well-formed formula) is discussed in Section 6.2 of [Sip], and is analogous to what we mean by a Boolean formula, where instead the variables take values in $\mathbb{N}$. For example $(x))$ is not a WFF (well-formed formula), since it has more right parentheses than left.

[2]For example, in the formula $\forall y, (x + y = y)$, y is quantified, but x is not; by contrast, $\exists x, \forall y, (x + y = y)$ is TQ (totally quantified), and is true for $\mathbb{N}$-valued variables provided that $\mathbb{N} = \{0, 1, 2, \ldots\}$, which is the convention in this article; it would be false for $\mathbb{N} = \{1, 2, 3, \ldots\}$.

Stronger forms of Gödel's Incompleteness Theorems actually explicitly construct such an "unprovable" statement; however, this depends on your method of "proving" statements, i.e., the specific axioms you use and specific rules you have of using your axioms to derive "proofs."

Even though we will not address methods of "proving" statements in this article (see why below), you should also be aware of some caveats to the last paragraph. First, the axioms you use have to be enumerated by some Turing machine, so there are either finitely many axioms or an infinite number that follow some pattern that can be specified by an algorithm. [Otherwise, if you were allowed to use any true statement in $\mathrm{Th}(\mathbb{N},+,\times)$ as an axiom, then you could trivially prove any theorem.] Second, we assume that any statement you "prove" is actually a true statement in $\mathrm{Th}(\mathbb{N},+,\times)$.

One can compare this to the way we stated Cantor's theorem: the weaker form states that for every map of sets $f\colon S\to \mathrm{Power}(S)$, f is not surjective; our stronger form says that $T=\{s\in S | s\notin f(s)\}$ is not in the image of f. Hence the stronger form explicitly produces a set not in the image of f.

We warn the reader that [Sip] adopts a few conventions that are not standard: [Sip] describes a statements to be a certain type of string (a WFF that is TQ) that can be written with the symbols

$$\big\{(,),\wedge,\vee,\neg,\forall,\exists,x,+,\times\big\}.$$

Most textbooks in logic have slightly different conventions. Although $(,),\wedge,\vee,\neg,\forall,\exists$ have their usual meaning,

(1) [Sip] uses the strings $x, xx, xxx,\dots$ to allow for an arbitrary (but finite!) number of variables when forming statements, and
(2) [Sip] uses $+,\times$ to denote operations $\mathbb{N}^3\to\{T,F\}$, namely $+(x,y,z)$ is true iff $x+y=z$, and similarly with $\times$.

In this article we follow the more common meaning of $+,\times$, i.e., as maps $\mathbb{N}^2\to\mathbb{N}$, and we use additional symbols $(<,=,\le)$ as maps $\mathbb{N}^2\to\{T,F\}$. Also we write our variables as $x_1,x_2,x_3,\dots$ which means that we add $0,\dots,9$ to the alphabet (similarly one can add $'$ to the alphabet and use the variables $x,x',x'',\dots$).

The main theorem stated in [Sip], namely Theorem 6.13, is that $\mathrm{Th}(\mathbb{N},+,\times)$ is undecidable, is due to Church, although largely based on ideas of Gödel, and, in particular, one ingenious trick/method. The reason we like this theorem is that it is perfectly precise. The proof of Theorem 6.13 of [Sip] relies on Lemma 6.14 there, which we state as follows.

Lemma 1.1. *For any Turing machine, M, and string w, there is a formula $\phi_{M,w}$ in the symbols*

$$\{+,\times,(,),\wedge,\vee,\neg,<,\forall,\exists,x_1,x_2,\dots\} \tag{1}$$

($x_1,x_2,\dots,x_n$ are called variables*) that contains a single free variable, x_1 (i.e., there is no $\forall$ or $\exists$ applied to x_1, but there is a $\forall$ or $\exists$ applied to every other variable), such that M accepts w iff the formula $\exists x_1,\phi_{M,w}$ is true when our variables take values in $\mathbb{N}$ (and $+,\times,<$ have their usual meaning, as described above).*

As claimed in [Sip], the idea is that x_1 will represent an accepting configuration of the Turing machine, M, on the input, w. However, beyond this, the proof of the above lemma is beyond the scope of [Sip]. The point of this article is to briefly

describe how $\phi_{M,w}$ is constructed. In fact, once you get used to a few simple ideas, there is only one (rather ingenious) trick/method.

1.1. **Consequence of Lemma 1.1.** Lemma 1.1 leads to the following theorem, which is clear from Section 6.2 of [Sip].

Theorem 1.2. *Let M be a Turing machine that recognizes some language, L, in the symbols (1) such that every string in L is a WF (well-formed) and TQ (totally-quatified) formula that is true in $\mathrm{Th}(\mathbb{N},+,\times)$. Then there is a WF and TQ formula, f, that is true in $\mathrm{Th}(\mathbb{N},+,\times)$ such that f is not accepted by M.*

Proof. Any WF and TQ formula, f, in the symbols (1) is either true or false in $\mathrm{Th}(\mathbb{N},+,\times)$. (You do have to prove this; you could do so by induction on the number of variables; once you prove that all quantifiers can be moved to the left of any ETC.) Hence for any such f, either f or $\neg f$ is true. If M recognized all true (WF and TQ) formulas, then $\mathrm{Th}(\mathbb{N},+,\times)$ would be decidable, contracting Lemma 1.1. □

1.2. **Why We Like Section 6.2 of [Sip] Lemma 1.1.** Theorem 6.12 and Lemma 6.13 of [Sip], and our Lemma 1.1 based on Section 6.2 of [Sip], are a terrific way to learn about incompleteness theorems because: (1) these statements are completely precise in the context of CPSC 421/501, and (2) it proves a weak form of incompleteness under the assumption that our definition of a "system of axioms and derivation rules" (or related notions) leads to a way of recognizing true statements with a Turing machine. Hence this serves as a clever intermediate point for the incompleteness theorems. Hence Section 6.2 of [Sip] serves to "factor" the proof of Gödel's Incompleteness Theorems into two parts: the part that we discuss in this article, and some extra reading (which requires a bunch of added technicalities). Furthermore to prove the results in Section 6.2 of [Sip] and Lemma 1.1 here, we need just a single clever idea.

1.3. $\mathbb{N}$ **and Enderton's Textbook** [End01]**.** In this article we use $\mathbb{N}$ to denote $\{0,1,2,\ldots\}$, rather than the usual $\{1,2,\ldots\}$. The main reason we do this is that this convention seems to be used in books on logic, such as Enderton's deservedly classic textbook [End01] (this is the 2nd edition). In this article we follow Enderton's textbook (currently available online free of cost via the UBC library to the UBC community).

We remark that CPSC 421/501 students (and instructors...) may be amazed at how much overlap there is between CPSC 421/501 and Chapter 1 of Enderton's textbook [End01]. For example, there Enderton defines a language to be *semidecidable*, just after Theorem 17E of Chapter 1, page 63, to mean we call *recognizable* in CPSC 421/501 and [Sip]. Chapter 1 of [End01] also discusses formulas and circuits.

1.4. **Gödel Numbers and the Proof of Lemma 1.1.** Many students may have heard of *Gödel numbers*. To define these, we denote the prime numbers with the notation:

$$p_0 = 2,\ p_1 = 3,\ p_2 = 5,\ p_3 = 7,\ \ldots$$

For any finite sequence of natural numbers, $a_0, a_1, \ldots, a_m \in \mathbb{N}$, we use

$$\langle a_0, \ldots, a_m \rangle_G = 2^{a_0+1} 3^{a_1+1} \ldots p_m^{a_m+1} \tag{2}$$

(see Chapter 3, page 220 of [End01]). Hence $\langle\ \rangle_G$ gives an injection $\mathbb{N}^* \to \mathbb{N}$ (it is not a surjection, since $3 = 2^0\,3^1$ is not in the image). There are, of course, minor variations of our formula (2) for Gödel numbers in the literature.

The general strategy to prove Lemma 1.1 is to express any finite number of configurations of the Turing machine M on input w as a sequence $a_0, \dots, a_m$ of finite length of some alphabet A, and to set

$$x_1 = \langle a_0, \dots, a_m \rangle_G = 2^{a_0+1} 3^{a_1+1} \dots p_m^{a_m+1}. \tag{3}$$

The simplest way to do this, given our proof of the Cook-Levin Theorem, is simply to take $A = \{T, F\}$ to use the Boolean variables $\{x_{ij\gamma}, y_{ij}, z_{iq}\}$ used in CPSC 421/501 (or the $\{x_{ijs}\}$ of [Sip]), arranged as a sequence $a_1, \dots, a_m$ (hence if the computation requires t steps, we need $m = O(t^2)$).[3]

Similarly to our proof of the Cook-Levin theorem, one now builds a sentence that expresses that x_1 represents a valid computation of our Turing machine. In other words, let Decode be any function $\mathbb{N} \times \mathbb{N} \to \mathbb{N}$ such that

$$\text{Decode}(2^{a_0+1} 3^{a_1+1} \dots p_m^{a_m+1}, b) = a_b \tag{4}$$

(since 0 can never be a Gödel number, Decode is not uniquely defined). All we need to do is build a sentence that checks that the values of $\text{Decode}(x_1, b)$ satisfy the Turing machine constraints. This is completely analogous to our proof of the Cook-Levin theorem; for a Turing machine computation that takes t steps, we will need to check $O(t^2)$ conditions between the values of $\text{Decode}(x_1, b)$ such that x_1 in (3) represents the unique valid computation to t steps. Hence we build a formula $\psi_{M,w,t}$ so that there is a computation accepting w within t steps iff M accepts w in t steps. Hence this is true iff $\exists x_1, \exists t, \psi_{M,w,t}$. Hence we take $\phi_{M,w,t}$ to be $\exists t, \psi_{M,w,t}$.

This proves Lemma 1.1 provided that we can prove the following.

Lemma 1.3 (Gödel)**.** *There exists a formula over* (1) *for a function* $\mathbb{N} \times \mathbb{N} \to \mathbb{N}$, $\text{Decode}(x, y)$, *that satisfies* (4).

Remark 1.4. To understand the lemma above, you need to understand how a formula can give a function $\mathbb{N} \to \mathbb{N}$. As an example, consider the *successor function*, $S\colon \mathbb{N} \to \mathbb{N}$ given by $S(x) = x + 1$. We have that $S(x) = y$ iff $x < y$ and there exists no z with $x < z$ and $z < y$. Hence, if x is any variable or expression, and we wish to use $S(x)$ in a formula, we can replace $S(x)$ by the formula

$$\forall y \Big((x < y) \wedge \big(\neg \exists z \big((x < z) \wedge (z < y) \big) \big) \Big) y \tag{5}$$

(we could also write $\exists y$ instead of $\forall y$). Hence a function $\mathbb{N} \to \mathbb{N}$ is really equivalent to expressing the phrase

$$(x < y) \wedge \big(\neg \exists z \big((x < z) \wedge (z < y) \big) \big) \tag{6}$$

which is really a function $\mathbb{N}^2 \to \{T, F\}$ taking (x, y) to T iff $y = S(x)$. Of course, (5) is obtained from (6) by adding $\forall y$ to the left ($\exists y$ would also work), and y to the right (and adding parenthesis, for clarity). Hence a formula for a function $\mathbb{N} \to \mathbb{N}$

[3] Alternatively, if $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{acc}}, q_{\text{rej}})$, one could take $A = \Gamma \cup \mathbb{N} \cup Q$, and use $a_1, \dots, a_m$ to express the tape cell contents, the tape head locations, and the states we are in at each step of the computation; this avoids the Boolean variables used in the Cook-Levin theorem, but it is still true that if the computation takes t steps, then we still need $m = O(t^2)$, since for each step we should keep track of the first t tape cells.

is equivalent to a formula for a function $\mathbb{N} \times \mathbb{N} \to \{T, F\}$. Similarly for functions $\mathbb{N}^m \to \mathbb{N}^{m'}$ and functions $\mathbb{N}^m \times \mathbb{N}^{m'} \to \{T, F\}$. (This point is touched on in [Sip], Example 6.11, page 254, where it is explained that the $+$ in $\mathrm{Th}(\mathbb{N}, +, \times)$ is, in [Sip], a relation $\mathbb{N}^3 \to \{T, F\}$ where $+(a, b, c)$ ([Sip] also uses $PLUS(a, b, c)$) is the function that is T iff $a + b = c$.)

This lemma is quite easy, except for one ingenious idea, needed to prove the following lemma.

Lemma 1.5 (Gödel)**.** *There exists a formula over* (1) *for a function Prime*: $\mathbb{N} \to \mathbb{N}$ *such that* $Prime(i) = p_i$*, the i-th prime (with* $p_0 = 2$*,* $p_1 = 3$*, etc.)*

1.5. **The Proof of Lemma 1.5.** We now divide the proof of Lemma 1.5 into two parts. First, we will expand the alphabet (1) to the larger alphabet

$$\{+, \times, (,), \wedge, \vee, \neg, \to, \leftrightarrow, \forall, =, <, \leq, \mathbf{0}, \mathbf{S}, \mathbf{E}, \exists, x_1, x_2, \ldots, x_n\}, \tag{7}$$

which includes some additional convenient symbols meaning:

(1) where $\to, \leftrightarrow$ have their usual logical meaning, i.e., $p \to q$ is $\neg p \vee q$ and $p \leftrightarrow q$ is $(p \to q) \wedge (q \to p)$;
(2) $=, <, \leq$ have their usual meaning in $\mathbb{N}$; e.g., $f \leq g$ means $\neg(g < f)$ and $f = g$ means $(f \leq g) \wedge (g \leq f)$ (we can similarly write $=, <, \leq$ over $\mathbb{N}$ in terms of any one of these three, e.g., $f \leq g$ means $\exists z(z + f = g)$),
(3) $\mathbf{0}$ is the symbol representing $0 \in \mathbb{N}$, which over $\mathbb{N}$ can be written by writing $\forall z, z + z = z$ (or $\exists z, z + z = z$) at the beginning of the formula, and then putting z in everywhere we want a $\mathbf{0}$;
(4) $\mathbf{S}$ is the successor function $x \mapsto x + 1$, e.g., $\mathbf{SS}x$ refer to $x + 2$; since $y = \mathbf{S}x$ is the unique element of $\mathbb{N}$ such that $x < y$ and there exists no z with $x < z$ and $z < y$, we can write $\mathbf{S}x$ where x is any variable or expression, by writing $\forall y((x < y) \wedge (\neg\exists z((x < z) \wedge (z < y))$, and putting y instead of $\mathbf{S}x$. (Do you see why $\mathbf{S}$ is convenient?)
(5) $\mathbf{E}$ is the exponentiation function, $x\mathbf{E}y$ refers to x^y. It is not at all obvious that we can express $\mathbf{E}$ in terms of any of string over $(,), \wedge, \vee, \neg, \forall, \exists, +, -, <, =, \leq$. To do so we will use a variant of the single ingenious trick.

We first prove:

Lemma 1.6. *Lemma 1.5 holds when* (1) *is replaced with the larger alphabet in* (7)*.*

This uses some easy facts and one ingenious idea that we will apply a number of times.

We will then use Lemma 1.5 using Lemma 1.6; the only non-trivial idea is how to express $\mathbf{E}$ with the symbols in (1). This can be done with the same ingenious idea as before.

The reader may want to think about how to prove Lemma 1.6, before reading ahead.

2. Some Easy Phrases and One Difficult Phrase

Let us take the symbols in (7), and write some phrases with them. It is quite easy to figure out how to write them down in succession. In these phrases x denotes either a variable or an expression.

(1) IsDivBy $=$ IsDivBy(x, y) a function $\mathbb{N}^2 \to \{T, F\}$ such that IsDivBy$(x, y) = T$ iff x divides y.

(2) IsPrime = IsPrime(x), the function $\mathbb{N} \to \{T, F\}$, which is true iff x is prime;
(3) NoPrimeBetween(x, y), a function $\mathbb{N}^2 \to \{T, F\}$ that equals T iff there is no prime number, z, with $x < z < y$;
(4) NextPrime = NextPrime(x), the function $\mathbb{N} \to \mathbb{N}$ that returns the smallest prime greater than x;
(5) PrimePower = PrimePower(x, y), the function $\mathbb{N}^2 \to \{T, F\}$ that is true iff x is a prime and y is a power of x (we will want to do this without using the symbol **E**);
(6) NextPrimePower = NextPrimePower(x, y), any function $\mathbb{N}^2 \to \mathbb{N}$ such that if x is prime, the function returns the smallest power of x greater than y (we will use PrimePower above, and again avoid using the symbol **E**).

We encourage the reader to construct these functions themself, before looking over our constructions below. Once you get the idea for one, the rest are not difficult.

(1) IsDivBy(x, y) can be written as $\exists z(x \times z = y)$.
(2) IsPrime = IsPrime(x) can be written as
$$(x < 2) \wedge \big(\forall z, \neg(1 < z < x) \vee \neg\text{IsDiv(z,x)}\big)$$
where $1 < z < x$ is shorthand for $(1 < z) \vee (z < x)$, and $1, 2$ are, respectively, shorthand for **S0**, **SS0**;
(3) NoPrimeBetween(x, y) is $\forall z, \neg(x < z < y) \vee \neg\text{IsPrime}(z)$.
(4) NextPrime(x) is
$$\forall y\big(\text{IsPrime}(y) \wedge \text{NoPrimeBetween}(x, y)\big)y$$

LATER IN THE ARTICLE:

an extremely clever ideahighly non-trivial to produce; they can all be done using essentially the same ingenious idea. Here we list a few.

(1) PrimeOfIndex: $\mathbb{N} \to \mathbb{N}$ given by PrimeOfIndex$(x) = p_x$, where $p_0 = 2, p_1 = 3, \ldots$
(2) SumOf(x, y, f), the sum from $x = 0$ to $x = y$ of an expression $f\colon \mathbb{N} \to \mathbb{N}$ (which presumably con...

3. Easy Observations Regarding Gödel Numbers and Sentences

We now express some functions and constants in $(\mathbb{N}, +, \times)$; these are not particularly surprising.

Our first constants and functions are implicit in [End01], where one is allowed to build senteces out of the symbols (see page 225 of Chapter 3, specifically Table XI there):

(1) Parameters: $\forall, \mathbf{0}, \mathbf{S}, <, +, \times, \mathbf{E}$ which are symbols numbered $0, 2, \ldots, 12$; and
(2) Logical symbols: $(,), \neg, \Rightarrow, =, v_1, v_2, \ldots$, which are symbols numbered $1, 3, 5, 7, \ldots$

Hence each symbol above is associated to a unique element of $\mathbb{N}$. The meaning of $\mathbf{0}, \mathbf{S}, \mathbf{E}$ are as follows: $\mathbf{0}$ refers to $0 \in \mathbb{N}$; $\mathbf{S}$ is the successor function, i.e., $\mathbf{S}x = x + 1$ (so $\mathbf{S}^2\mathbf{0} = \mathbf{SS0}$ refers to $2 \in \mathbb{N}$); and $\mathbf{E}$ is the exponentiation function, i.e. $x\mathbf{E}y$ refers to x^y. The variables $v_1, v_2, \ldots$ are any (finite set of variables); we will often use other letters $(x, y, z, \ldots$ or $a, b, c, \ldots)$ to denote these variables.

The first task is to express the above in terms of $(\mathbb{N}, +, \times)$, meaning sentences with variables, and the symbols $+, \times$, and the usual symbols of *first-order logic*, i.e., $\exists, \forall, \neg, \wedge, \vee, \Rightarrow, =$ and commas and parentheses. Of course, $p \Rightarrow q$ is equivalent to $\neg p \vee q$, so that some of these symbols are redundant; moreover, it is often convenient to use some other symbols, like $\iff$ (where $p \iff q$ is equivalent to writing $(p \Rightarrow q) \wedge (q \Rightarrow p)$ or $(p \wedge q) \vee (\neg p \wedge \neg q)$. Let us describe some of the symbols

(1) The function $a < b$, viewed as a function $\mathbb{N} \times \mathbb{N} \to \{T, F\}$ (representing true, false) can be expressed by

$$\exists c \big((a + c = b) \wedge \neg (c + c = c) \big)$$

(here $\exists c$ means $\exists c \in \mathbb{N}$; all quantifiers $\exists$, $\forall$ refer to variables that lie in $\mathbb{N}$). In other words, $a < b$ means there exists a c such that $a + c = b$ and c is not 0 (equivalent to $\neg(c + c = c)$).

(2) the integer 0 can be expressed as the constant a,

$$\exists a \, (a + a = a),$$

hence one can place this in the beginning of any sententence to make a refer to $\mathbf{0}$;

(3) the function $\mathbf{S}$ can be described by $\mathbf{S}x$ refers to the unique y satisfying

$$\exists y \Big((x < y) \wedge \big(\neg \exists z, \, (x < z) \wedge (z < y) \big) \Big).$$

Hence instead of $\mathbf{S}x$ we write the above phrase, followed by substituting y whenever we write $\mathbf{S}x$.

After doing this with all symbols, we introduce some not particularly surprising phrases, such as

(1) The function $\text{IsDivBy}(x, y)$ is the map $\mathbb{Z}^2 \to \{T, F\}$ which is T iff x divdes y; we can write $\text{IsDivBy}(x, y)$ as

$$\exists z \, (\mathbf{0} < z) \wedge (x \times z = y);$$

(2) The function $\text{IsPrime}(x)$ is the map $\mathbb{Z} \to \{T, F\}$ which is T iff x is prime: $\text{IsPrime}(x)$ can be written as

$$\neg \exists y \, (\mathbf{S0} < y) \wedge (y < x) \wedge \text{IsDivBy}(y, x);$$

(3) The function $\text{NextPrime}(x, y)$ is true if $x < y$ are both prime, and there is no prime bewteen x, y:

$$(x < y) \wedge \text{IsPrime(x)} \wedge \text{IsPrime}(y) \wedge \Big(\neg \exists \, z, \text{IsPrime(z)} \wedge (x < z) \wedge (z < y) \Big).$$

4. Some Surprising Sentences

Here are some sentences that I find surprising and extremely clever, beginning in item 7., page 219, Chapter 3 of [End01]:

(1) One can express the function $x\mathbf{E}y$ by a phrase in $(\mathbb{N}, +, \times)$; to do so it suffices to give a phrase for the function $f \colon \mathbb{Z}^3 \to \{T, F\}$ such that $f(x, y, z) = T$ iff $z = x^y$, and then to write $\exists z, f(x, y, z)$, and put z wherever you want to put $x\mathbf{E}y$. But how can you express $f(x, y, z)$ with only $+, \times$ and the usual operations $\forall, \exists, \neg, \wedge, \vee$ of first-order logic??

(2) One can express the function $g\colon \mathbb{N} \to \mathbb{N}$ such that $g(a) = p_a$, i.e., the a-th prime (counting from $2 = p_0$ and $3 = p_1$). Again, it suffices to express the function $f\colon \mathbb{N}^2 \to \{T, F\}$ such that $f(a, b) = T$ iff iff $b = p_a$, i.e., b is the a-th prime. But how to do this in $+, \times$ plus first-order logic symbols??
(3) One can express the function (m, b) such that if $m = \langle a_1, \dots, a_N \rangle_{\mathrm{G}}$, then the function returns $c = a_b$ if $b \le N$, and gives a non-existent variable c if $b > N$. You may wish to try to build these sentences yourself before seeing how they are done.

5. The Construction of Some Surprising Sentences

Enderton's textbook [End01] really comes to life on page 219 in Chapter 3, with the following ingenious idea.

Lemma 5.1. *Let $a, b \in \mathbb{N}$. Then the following are equivalent:*

$b = p_a$ (where $p_0 = 2$, $p_1 = 3$, etc.); andthere exists a $c \in \mathbb{N}$ such that c satisfies:

(2) *(a) c is odd, i.e., $\mathrm{IsDivBy}(2, c) = F$;*
(b) $c \le b^{(a^2)}$ (one can alternatively write $c \le b^{(b^2)}$);
(c) for every $j \in \mathbb{N}$, and prime numbers q, r such that $r \le b$, and $\mathrm{NextPrime}(q, r) = T$, we have

$$\forall j, \big(q^j \text{ is divisible by } c \iff r^{j+1} \text{ is divisible by } c\big);$$

and
(d) b^a divides c and b^{a+1} does not divide c.

Proof. (1)⇒(2): we immediately see that

$$c = 2^0\, 3^1\, 5^2\, \dots\, p_a^a$$

satisfies (2)(a,b,c,d).

(2)⇒(1): let c exist satisfying (2)(a,b,c,d).*Let*$c = 2^{i_0} 3^{i_1} \cdots p_a^{i_a} N$,*where*N*is free of prime factors*2,3,...,p_a. We easily prove by induction that $i_j = j$ for all $0 \le j \le a$: indeed, the case $a = 0$ holds since c is odd; if this claim holds for some value of $a \in \mathbb{N}$, then we have $i_a = a$, and hence (2c) implies that $i_{a+1} = a + 1$. Then (2d) implies that $b = p_a$. □

Notice that 2(b) is not essential to the proof. However, 2(b) is presumably useful in that the search for c can be limited to a finite range.

Lemma 5.2. *Let $b \in \mathbb{N}$ be prime, and let $b = p_a$ (i.e., the a-th prime, with $p_0 = 2$, $p_1 = 3$, etc.). For $c \in \mathbb{N}$, the following are equivalent:*

(1)

$$c = 2^0\, 3^1\, 5^2\, \dots\, p_a^a, \tag{8}$$

and

(2) c satisfies:
(a) c is odd, i.e., $\mathrm{IsDivBy}(2, c) = F$;
(b) $c \le b^{(a^2)}$ (one can alternatively write $c \le b^{(b^2)}$);
(c) for every $j \in \mathbb{N}$, and prime numbers q, r such that $r \le b$, and $\mathrm{NextPrime}(q, r) = T$, we have

$$\forall j, \big(q^j \text{ is divisible by } c \iff r^{j+1} \text{ is divisible by } c\big);$$

(d) b^a *divides* c *and* b^{a+1} *does not divide* c*; and*
(e) if $b < r$ *and* r *is prime, then* r *does not divide* c*.*

Proof. (1)⇒ (2) is clear. So let $c \in \mathbb{N}$ satisfy (2). Hence $c \geq 1$ is odd; if $b = 2$, then $a = 0$, etc. □

---

(1) Clever(a) takes an $a \in \mathbb{N}$ and returns the integer

$$c = 2^0\, 3^1\, 5^2\, \ldots\, p_a^a;$$

this can be done by noting that

---

(2) The function $p_a = b$, i.e., b is the a-th prime number, can be expressed as follows: first note for any $a \in \mathbb{N}$, setting

$$b = p_a^a, \quad c = 2^0\, 3^1\, 5^2\, \ldots\, p_a^a, \tag{9}$$

we have
(a) $c \leq b^{a^2}$ and 2 does not divide c;
(b) for all q, r such that NextPrime(q, r) and $r \leq b$ are true, we have

$$\forall j, \big(q^j \text{ is divisible by } c \iff r^{j+1} \text{ is divisible by } c\big);$$

and
(c) b^a divides c and b^{a+1} does not divide c.
Conversely, note that if $a \in \mathbb{N}$, then if b, c satisfy for b, c we have $b = p_a$ and $c = 2^0 3^1 \ldots p_a^a$ and b is prime

---

the above three conditions determine c from a, b.

---

(3) The phrase $\langle a_0, \ldots, a_m \rangle_{\mathrm{G}} = b$ can be expressed as follows:
(4)

6. The Ingenious Idea as a "For Loop," and Its Consequences

We claim the ingenious trick to prove Lemma 1.3 is equivalent to being able to write a "for loop." In this section we explain this, and derive a number of consequences.

---

7. Leftover

which roughly speaking says that in working with sentences over $(\mathbb{N}, +, \times)$ plus the usual symbols of first order logic (namely $\forall, \exists, \neg, \wedge, \vee$),

References

[End01] Herbert B. Enderton, *A mathematical introduction to logic*, second ed., Harcourt/Academic Press, Burlington, MA, 2001. MR 1801397

Department of Computer Science, University of British Columbia, Vancouver, BC V6T 1Z4, CANADA.
Email address: jf@cs.ubc.ca
URL: http://www.cs.ubc.ca/~jf