1. Gödel's first incompleteness theorem

We offer here a naïve presentation of Gödel's first incompleteness theorem that relies on recursion theoretic techniques. It says that any "reasonable" axiomatic system that attempts to have as theorems precisely all the "true" (first order) formulas of arithmetic will fail: There will be infinitely many true formulas that are not theorems. The qualifier "reasonable" could well be replaced by "practical": One must be able to tell, algorithmically, whether a formula is an axiom or not—for how else can one check a proof, let alone write one? "True" means true in the standard interpretation over the structure

$$\mathfrak{N} = (\mathbb{N}; 0, \lambda x.x + 1, \lambda xy.x + y, \lambda xy.x \times y, \lambda xy.x < y)$$

where after the semicolon I have listed "important" constants, functions and predicates (see the table of interpretations below).

To set the stage more accurately, we will need some definitions and notation. In order to do arithmetic we need, first of all, a first order (logical) language which we use to write down formulas and proofs. Before we get to the language (that is, a set of strings) we need an "alphabet".

This alphabet has two parts: One that is standard in all such languages (*logical symbols*), namely variables of natural number type (or "object variables") *denoted by* x, y, z, u, v, w with or without accents and subscripts. The logical part also contains the symbols

$$\neg, \vee, \wedge, \rightarrow, \leftrightarrow, =, \exists, \forall, (,)$$

The other part of the alphabet is specific to doing arithmetic. It contains the "abstract" symbols

$$0, S, +, \times, <$$

These are the *nonlogical symbols* which we can interpret any way we please. *However we decide to interpret them in the standard way*:

Abstract (language) symbol	**Concrete interpretation**
0	0 (zero)
S	$\lambda x.x + 1$
$+$	$\lambda xy.x + y$
$\times$	$\lambda xy.x \times y$
$<$	$\lambda xy.x < y$

One forms formulas now in the standard way (see for example [Tou98]). But what about symbols for "1", "2" and "112056734555"? An arbitrary $n \in \mathbb{N}$ has the formal counterpart

$$\underbrace{SS \cdots S}_{n \text{ times}} 0$$

which we denote by $\widetilde{n}$.

Thus, an axiomatic system (or theory) that we use to formally (i.e., syntactically) prove theorems of arithmetic contains:

(1) The briefly described first order language above.[*]

(2) A recursive set of strings: The well-formed formulas (wff).[†]

(3) A recursive subset of wff: The special (or nonlogical) axioms for arithmetic.[‡]

(4) The logical axioms (cf. [Tou98])

(5) The rules of inference (cf. [Tou98])

1.1 Remark.

(i) Intuitively, (2) above says that there is an algorithm that for every string over our alphabet

$$\neg, \vee, \wedge, \rightarrow, \leftrightarrow, =, \exists, \forall, (,), v, 1, 0, S, +, \times, < \qquad (A)$$

will decide the string's membership in wff—i.e., whether or not the string parses correctly as a formula. But what do we mean when we say that a set of *strings* is recursive? After all, until now all our recursive sets were sets of numbers (or of tuples of numbers). Well, nothing has changed! Imagine that the symbols in our alphabet (A) above, in precisely the order given, are just (strange) symbols for the numbers 1 through 17. Then *any* string over the alphabet *denotes a number* base 18[§] (if you buy hexadecimal numbers like BBC—i.e., 3004 in decimal—then you have to buy what I have just said). For example, $(\exists v1v)v1v = 0$ denotes (in decimal notation) 33651680537461.

Thus, to speak of a set of strings over (A) is the same as talking about a subset of $\mathbb{N}$.

(ii) We have said right at the beginning that the set of axioms must be "reasonable", i.e., "recognizable". That is precisely what item (3) above asks for in stipulating "recursive". But don't we want to be able to recognize *all*

[*] As we know from MATH 1090, first order means that we are only allowed to quantify object variables, i.e., write things such as $\exists x$ and $\forall x$. We are not allowed to write "for some (all) predicates" for example.

[†] It is easy to do the following: (a) Generate all object variables—what we casually called by the *meta* names x, y, x'', z'''_{104}, etc—by the regular expression $v1^*v$ over $\{v, 1\}$. (b) Carefully define syntax of formulas so that the language is context-free. It then follows (cf. COSC 2001) that the language—the set of all wff—is recursive.

[‡] A particularly famous choice of axioms is due to Peano—the so-called Peano Arithmetic, for short "PA". It has axioms that give the behaviour of every nonlogical symbol, plus the induction "axiom schema":

$$P(0) \wedge (\forall x)(P(x) \rightarrow P(Sx)) \rightarrow (\forall x)P(x)$$

The "schema" (or "form", or "template" in English) gives one axiom for each choice of wff P.

[§] Why not number the symbols in (A) by 0 through 16 and work base 17? Because we will have trouble with things like $\neg 0 < 0$. The "digit" in the most significant position is (of value) 0 and we lose information as we pass to numerical value. I.e., both $\neg 0 < 0$ and $0 < 0$ denote the same number.

the axioms, including the logical ones? Yes, but these do form a recursive (recognizable) set. You can easily check so after you refresh your memory on which *are* the logical axioms (cf. [Tou98]).

It is also worth stating that the rules of inference are algorithmically applicable. For example, if we are to use the rules in [GS94], then they are all algorithmic—easily. Take "equanimity"

$$\text{if } A \leftrightarrow B \text{ and } A \text{ then } B$$

Whether it applies or not is determined by its *form*. □

1.2 Lemma. (Informal) *The set of all theorems of an axiomatic arithmetic as this has been described in* (1)–(5) *above is r.e.*

Proof. (**Informal**) Let us add one new symbol, # to the alphabet (A), to form the augmented alphabet

$$\neg, \vee, \wedge, \rightarrow, \leftrightarrow, =, \exists, \forall, (,), v, 1, 0, S, +, \times, <, \# \tag{B}$$

We use # as "glue" to concatenate all the formulas in any given proof that we may write. I.e., we explicitly write # once between every pair of consecutive formulas

$$F_1, F_2, \ldots, F_n$$

of a proof. We thus convert it to a *single string* (over (B))

$$F_1 \# F_2 \# \ldots \# F_n \tag{iii}$$

Giving "digit value" to # equal to 18, we can think of any expression (string) like (iii) above as a number notation base 19.

We now simultaneously and algorithmically generate two lists, L_1 and L_2.

L_1 consists of *all* strings over (B), enumerated by increasing length and in each length group lexicographically.

For every string S enumerated thus, we test (algorithmically—being able to tell an axiom from a non axiom helps here!) whether or not it has the form (iii) *and* is the glued form of a proof. If the test succeeds, then we add to list L_2 the "tail" of S that follows the rightmost # in S (if S is exactly the sequence in (iii), then the tail is F_n).

This process lists algorithmically, under L_2, all the theorems of our theory. I.e., the theorem set is r.e. □

Let us call **Complete Arithmetic**, for short CA, the set

$$\{S : S \text{ is a wff that is true when interpreted in } \mathfrak{N} \text{ as per table on p.1}\}$$

We now have:

1.3 Gödel's First Incompleteness Theorem. *Every axiomatic system for arithmetic that satisfies* (1)–(5) *above is* incomplete *in the sense that its set of theorems cannot equal the set CA.*

Proof. In view of the preceding lemma, it suffices to prove that CA is not r.e.

To this end, consider the two sets of wffs below:

$$\widetilde{K} = \{\phi_{\widetilde{a}}(\widetilde{a}) \uparrow : \phi_{\widetilde{a}}(\widetilde{a}) \uparrow \text{ is true as interpreted in } \mathfrak{N}\}$$

and

$$Q = \{\phi_{\widetilde{a}}(\widetilde{a}) \uparrow : a \in \mathbb{N}\} \tag{1}$$

Before we proceed, let me note that "$\phi_{\widetilde{a}}(\widetilde{a}) \uparrow$" is an abbreviation of a formula of arithmetic—in other words, it can be written down, in principle, using no more than the symbols from the alphabet (A), and this wff will "formally say", precisely, $\phi_a(a) \uparrow$ (see Appendix).

Thus,

$$\widetilde{K} = \{\phi_{\widetilde{a}}(\widetilde{a}) \uparrow : a \in \overline{K}\} \tag{2}$$

Now, given any $a \in \mathbb{N}$, I can construct the formula $\phi_{\widetilde{a}}(\widetilde{a}) \uparrow$—or, using a different abbreviation, $\neg(\exists y)T(\widetilde{a}, \widetilde{a}, y)$. That is, viewing strings as numbers, the function f that on input a outputs (the number denoted by the string abbreviated by) $\phi_{\widetilde{a}}(\widetilde{a}) \uparrow$ is, intuitively at least, recursive.

Thus

$$\overline{K} \le \widetilde{K} \tag{3}$$

by (2), since (2) says $a \in \overline{K}$ iff $f(a) \in \widetilde{K}$. On the other hand,

$$Q = \mathrm{ran}(f) \tag{4}$$

by (1). Thus, $\widetilde{K}$ is *not* r.e., while Q *is* r.e.

Note that

$$\mathrm{CA} \cap Q = \widetilde{K}$$

thus CA cannot be r.e. by closure properties. □

Note the emphasized "every" in the theorem. It draws attention to the fact that we have not *fixed* any *particular* "reasonable" theory—whatever we said holds for all recursive theories. "Every" moreover implies that every theory that the theorem speaks of misses an infinite chunk of CA. Indeed, if it only misses a finite chunk, C, then adding the formulas of C as axioms we still get a recursive set of axioms (closure properties, and the fact that finite sets are recursive). But this new set covers all of CA (why?) contrary to Gödel's theorem.

2. Appendix: $\phi_{\widetilde{a}}(\widetilde{a}) \uparrow$ is a formula of arithmetic

To show that $\phi_{\widetilde{a}}(\widetilde{a}) \uparrow$ abbreviates a formula of arithmetic, it suffices to prove that there is a formula of arithmetic that expresses the Kleene predicate $T(x, y, z)$.

It will so follow if we can prove that for every function $f \in \mathcal{PR}$ its graph is expressible arithmetically, for then if c_T is the characteristic function of T, $c_T(x,y,z) = w$—and therefore $c_T(x,y,z) = 0$—can be written down as a formula of our language of arithmetic.

To make life easy we add one more binary function symbol to the language—say, "@"—that we interpret as $\lambda xy.x^y$. This will allow us—when the time comes in the present argument—to continue using prime power coding as we have been doing all along.[¶]

2.1 Lemma. *The following relations can be written in our formal language.*

1. $z = x \dot{-} y$
2. $x \,|\, y$
3. $Pr(x)$
4. $Seq(z)$
5. $Next(x,y)$ *(meaning $x < y$ are consecutive primes)*
6. $pow(z,x,y)$ *(meaning $x > 1$ and y is the highest power of x dividing z)*
7. $W(z)$ *(meaning z has the form $p_0 p_1^2 p_2^3 \cdots p_n^{n+1}$ for some n)*
8. $y = p_n$
9. $z = \exp(x,y)$

If we keep a keen eye on the task at hand we realize that we do not worry about bounding our quantifications, for it is not our purpose to show these relations in $\mathcal{PR}_*$. Indeed we know from earlier work that they are in this set. This time we want to show that we can express them within our language, which allows us to write down, *in the first instance*, only the symbols for 0, successor, addition, multiplication, exponentiation (added in this section) and "less than". We want to show that our language is rich enough to allow us write down a lot of other things, including all of the above.

Proof.

1. $z = x \dot{-} y$: This is expressed by $z = 0 \wedge x < y \vee x = z + y$
2. $x \,|\, y$: This is expressed by $(\exists z) y = xz$ (we are using "implied multiplication" throughout: "xy" rather than the formally correct "$x \times y$")
3. $Pr(x)$: This is expressed by $x > 1 \wedge (\forall y)(y \,|\, x \rightarrow y = 1 \vee y = x)$[‖]

[¶]Using the so-called Chinese Remainder Theorem of number theory Gödel effected the coding with his "beta function" *not* needing to add exponentiation to the system.

[‖]If we wanted to be pedantic we would have written "$S0$" or $\widetilde{1}$" instead of "1".

4. $Seq(z)$: This is expressed by

$$z > 1 \wedge (\forall x)(\forall y)(Pr(x) \wedge Pr(y) \wedge x < y \wedge y \mid z \to x \mid z)$$

5. $Next(x, y)$: This is expressed by

$$Pr(x) \wedge Pr(y) \wedge x < y \wedge \neg(\exists z)(Pr(z) \wedge x < z \wedge z < y)$$

6. $pow(z, x, y)$: This is expressed by $x > 1 \wedge x^y \mid z \wedge \neg x^{y+1} \mid z$**

7. $W(z)$: This is expressed by

$$\begin{aligned}&Seq(z) \wedge \neg 4 \mid z \wedge \\ &\quad (\forall x)(\forall y)\big(Next(x, y) \wedge y \mid z \to (\exists w)(pow(z, x, w) \wedge pow(z, y, w + 1))\big)\end{aligned}$$

8. $y = p_n$: This is expressed by $(\exists z)(W(z) \wedge pow(z, y, n + 1))$

9. $z = \exp(x, y)$: This is expressed by

$$(\exists w)(pow(y, w, z) \wedge w = p_x)$$

$\square$

We can now prove the following theorem that concludes the business of this section.

2.2 Theorem. *For every $f \in \mathcal{PR}$, its graph $y = f(\vec{x}_n)$ is expressible in our language of arithmetic (that now includes—as you recall—a symbol for exponentiation).*

Proof. We do induction on $\mathcal{PR}$:

(1) **Basis.** There are three graphs to worry about here: $y = x + 1$, $y = 0$ and $y = x$ (or, fancily, $y = x_i$; or more fancily $y = u_i^n(\vec{x}_n)$). Clearly, all are expressible in our language (e.g., the first looks somewhat different in the formal notation: $y = Sx$; the other two look the same whether formally or informally).

(2) **Composition.** Say, the property is true for the graphs of $f, g_1, \ldots, g_n$. This is the induction hypothesis (I.H.)

How about $y = f(g_1(\vec{x}_m), g_2(\vec{x}_m), \ldots, g_n(\vec{x}_m))$? Well, this graph is expressible as

$$(\exists u_1) \cdots (\exists u_n)\big(y = f(\vec{u}_n) \wedge u_1 = g_1(\vec{x}_m) \wedge \cdots \wedge u_n = g_n(\vec{x}_m)\big)$$

and we are done by the I.H.

**We wrote "$y+1$" rather than the formal "Sy". By the way, in defining *pow* we benefitted from the presence of exponentiation.

(3) **Primitive recursion.** This is the part that needs the previous lemma. Here's why: Assume (I.H.) that the graphs of h and g are expressible, and let f be given for all $x, \vec{y}$ by

$$\begin{aligned} f(0, \vec{y}) &= h(\vec{y}) \\ f(x+1, \vec{y}) &= g(x, \vec{y}, f(x, \vec{y})) \end{aligned}$$

Now, to say $z = f(x, \vec{y})$ is equivalent to saying

$$\begin{aligned} (\exists m_0)(\exists m_1)\cdots(\exists m_x)\Big(&m_0 = h(\vec{y}) \wedge z = m_x \\ &\wedge (\forall w)\big(w < x \to m_{w+1} = g(w, \vec{y}, m_w)\big)\Big) \end{aligned} \tag{i}$$

The trouble with the "formula" (i) above is that it is not a formula at all, because it has a variable length prefix: $(\exists m_0)(\exists m_1)\cdots(\exists m_x)$. Coding to the rescue! Let us use a single number,[††]

$$m = p_0^{m_0} p_1^{m_1} \cdots p_x^{m_x}$$

to represent all the m_i, for $i = 0, \ldots, x$. Clearly,

$$m_i = \exp(i, m), \text{ for } i = 0, \ldots, x$$

We can now rewrite (i) as

$$\begin{aligned} (\exists m)\Big(&\exp(0, m) = h(\vec{y}) \wedge z = \exp(x, m) \\ &\wedge (\forall w)\big(w < x \to \exp(w+1, m) = g(w, \vec{y}, \exp(w, m))\big)\Big) \end{aligned} \tag{ii}$$

The above is a formula of arithmetic because of the I.H. Some parts of it are a bit cryptic. For example, the part

$$\exp(w+1, m) = g(w, \vec{y}, \exp(w, m))$$

can be expressed in more detail as

$$(\exists u)(\exists v)\big(u = \exp(w+1, m) \wedge v = \exp(w, m) \wedge u = g(w, \vec{y}, v)\big)$$

The above is expressible by the I.H. and the preceding Lemma. This completes the proof. $\square$

[††] We do not need the more fancy $\langle m_0, \ldots, m_x \rangle$ here.

Bibliography

[GS94] David Gries and Fred B. Schneider, *A Logical Approach to Discrete Math*, Springer-Verlag, New York, 1994.

[Tou98] G. Tourlakis, *A basic formal equational predicate logic*, Technical Report CS-1998-09, York University, Dept. of Comp. Sci., 1998.