Beautifying Gödel

Eric C.R. Hehner
University of Toronto

Introduction

The incompleteness theorems of Kurt Gödel [1931] are considered to be among the most important results of mathematics. They are regarded as "deep", and they strongly influenced the course of modern mathematics. They are popularly thought to prove the limitations (or even futility!) of mathematical formalism. At any rate, they deserve to be presented as simply, as elegantly, as beautifully as possible. Gödel's own presentation was careful and clear, but not nearly as simple as it could be.

Our beautification is made in stages. It is tempting to skip the stages and present just the final form: a three line proof. Unfortunately the stages are necessary to convince the reader that the essence of Gödel's theorems has not been lost. And there are some lessons to learn along the way. The theorems belong to the branch of mathematics known as metamathematics, so we start there.

Metamathematics: The Study of Formalisms

To study the stars, it is helpful to design a mathematical formalism for the purpose. Mathematics is not limited, however, to the study of nature. It also helps us to study the artificial worlds of bridges, economics, music, and even thought processes. And if we want to make mathematical formalisms objects of study, it is helpful to design a mathematical formalism for the purpose.

We design a formalism (synonymously, a theory) so that its sentences represent statements about the objects of study. Some of these statements are true, and others are false; we design the theory so that its theorems (provable sentences) represent true statements and its antitheorems (disprovable sentences) represent false statements. Suppose, for example, that the object of study is a version of number theory; let us call it NT . Here are eight examples of true statements about (not in) NT .

(a) $1+1=2$ is a theorem of NT .
(b) $1+1=3$ is an antitheorem of NT .
(c) $0÷0=4$ is neither a theorem nor an antitheorem of NT .
(d) $(0÷0=4) \vee \neg(0÷0=4)$ is a theorem of NT .
(e) A sentence is an antitheorem of NT if and only if its negation is a theorem of NT .
(f) A sentence is a theorem of NT if and only if its negation is an antitheorem of NT .
(g) No sentence is both a theorem and an antitheorem of NT .
(h) Every sentence of the form $s\vee\neg s$ is a theorem of NT .

Here are six false statements about NT .

(i) $1+1=2$ is an antitheorem of NT .
(j) $1+1=3$ is a theorem of NT .
(k) $0÷0=4$ is either a theorem or an antitheorem of NT .
(l) A sentence is an antitheorem if and only if it is not a theorem of NT.
(m) A sentence is either a theorem or an antitheorem of NT .
(n) Either a sentence or its negation is a theorem of NT .

Statement (a) shows us a simple theorem. In mathematics texts, it saves space to write a sentence such as 1+1=2 without saying anything about it, thereby meaning that it is a theorem. But in this paper we shall not do so. Statement (g) says that NT is consistent. The false statement (m) says that NT is complete. We included division in NT to give a simple sentence that is neither a theorem nor an antitheorem; Gödel's surprise result was that, even without division, with only addition and multiplication, there are sentences that are neither theorems nor antitheorems.

Let us call our theory to study theories TT . For any theory T and sentence *s* of T we introduce the sentence (of TT)

$$\mathsf{T}\vdash s$$

to represent the (true or false) statement that *s* is a theorem of T . And we introduce

$$\mathsf{T}\dashv s$$

to represent the statement that *s* is an antitheorem of T . When it is clear which theory is under study, we may omit its name and write simply $\vdash s$ and $\dashv s$. Because of statement (e) we may be tempted to dispense with the symbol for “antitheorem”, and to speak instead of the negation of a theorem. However, it is not necessary for all theories to include negation, and it may be interesting to study some that do not. Because of (c) we certainly cannot take “is an antitheorem” to mean “is not a theorem”. We need a symbol for “antitheorem” for the same reason that boolean algebra needs symbols for both “true” and “false”, and indeed “true” and “false” are a primitive theorem and antitheorem respectively. We also need symbols for “or”, “and”, “not”, “if and only if”, and quantifiers, as seen by our example statements. To avoid confusion, we might insist (as did Kleene) that the symbols of theory TT differ from those of the theories under study. But we may want to use TT to study TT as well, and as we see in the next paragraph, there is a better way to avoid confusion. We therefore reuse $\lor$, $\land$, $\neg$, $=$, $\forall$, and $\exists$ in TT .

The way to avoid confusion was known to Gödel and is well-known to programmers: it is to distinguish program from data. A compiler writer knows the difference between her program and her data, even though her data is someone else's program, even if it is in the same language. To her, the incoming data is a character string, and her program examines its characters. Similarly in metamathematics one theory can describe another without confusion, even if that other theory is itself, by realising that, to the describing theory, expressions of the described theory are data of type character string.

In modern logic, the distinction between program and data is not always made, and $\vdash$ is applied directly to sentences. To partially compensate, logicians distinguish between “extensional” and “intensional” operators, and make rules stating when something cannot be substituted for its equal. For the sake of simplicity and clarity, let us maintain the programmer's distinction: we apply $\vdash$ to a character string representing a sentence. Thus

$$\mathsf{NT}\vdash \text{“1+1=2”}$$

is the sentence of TT representing statement (a).

Omitting the name NT , the statements (a) to (n) are represented (formalized) in TT as follows. (Juxtaposition of character strings indicates (con)catenation; quantification is over character strings.)

(aa) $\vdash$ “1+1=2”
(bb) $\dashv$ “1+1=3”
(cc) $\neg\vdash$ “0÷0=4” $\land\ \neg\dashv$ “0÷0=4”
(dd) $\vdash$ “(0÷0=4) ∨ ¬(0÷0=4)”

(ee) $\forall s\cdot \dashv s \;=\; \vdash (\text{“}\neg\text{”}\ s)$
(ff) $\forall s\cdot \vdash s \;=\; \dashv (\text{“}\neg\text{”}\ s)$
(gg) $\neg\exists s\cdot \vdash s \land \dashv s$
(hh) $\forall s\cdot \vdash (s\ \text{“}\lor \neg\text{”}\ s)$

(ii) $\dashv$ “1+1=2”
(jj) $\vdash$ “1+1=3”
(kk) $\vdash$ “0÷0=4” $\lor$ $\dashv$ “0÷0=4”
(ll) $\forall s\cdot \dashv s \;=\; \neg \vdash s$
(mm) $\forall s\cdot \vdash s \lor \dashv s$
(nn) $\forall s\cdot \vdash s \lor \vdash (\text{“}\neg\text{”}\ s)$

We try to design TT so that (aa) to (hh) are theorems, and (ii) to (nn) are antitheorems. When we design a theory to study the stars, we should always retain some doubt about how well our theory matches the facts. The same goes for a theory to study theories.

Classical and Constructive Mathematics

In the previous section, statements (h) and (n) are very similar, but (h) is true and (n) false. Which of them is the Law of the Excluded Middle? The truth of this “law” was hotly disputed for a while by mathematicians who conducted their mathematics informally; perhaps the informality was necessary to the dispute. For a formal theory like NT, we can distinguish (h), which is the “law”, from (n), which (given (e)) is a statement of completeness.

There are interesting theories for which the Law of the Excluded Middle does not hold; instead a proof of a disjunction requires a proof of one of the disjuncts. We may represent this in TT as follows.

$$\forall s\cdot \forall t\cdot \vdash(s\ \text{“}\lor\text{”}\ t) \;=\; \vdash s \lor \vdash t$$

These theories are called “constructive”. Similarly a proof of $\exists x\cdot p\ x$ requires a term t such that $p\ t$ is a theorem. A proof of $\forall x\cdot \exists y\cdot p\ x\ y$ is a program for constructing from input x a satisfactory output y.

Theories in which the Law of the Excluded Middle holds are called “classical”. Most people are willing to agree that “either God exists or God does not exist” without a proof of either disjunct. These people prefer a classical theory in which

$$(0\div 0=4) \;\lor\; \neg(0\div 0=4)$$

is a theorem even though neither disjunct is a theorem. In the next section we prove the equality of two sentences even though neither is a theorem nor an antitheorem.

The preceding discussion conceals a subtle point. We have said that in a classical theory, a disjunction may be a theorem even though neither of its disjuncts is. If our metatheory TT is classical, then perhaps $\vdash s \lor \vdash t$ can be a theorem (for some choice of s and t) even though neither of its disjuncts is. If so, then the TT sentence

$$\forall s\cdot \forall t\cdot \vdash(s\ \text{“}\lor\text{”}\ t) \;=\; \vdash s \lor \vdash t$$

does not represent our intention to describe a constructive theory. For this reason, we may prefer our metatheory to be constructive. Unfortunately, if the metatheory is used to describe itself, it cannot tell us whether it has this constructive property.

Gödel's First Incompleteness Theorem

Very roughly, Gödel's argument goes as follows. He first created an elaborate scheme to encode sentences as numbers. He then created a sentence which, on the surface, concerns natural numbers, but which may be viewed as code for "I am not a theorem". The "I" in that sentence is the number that encodes that very sentence. It is the Liar's Paradox in a new suit. If this sentence is a theorem, then it is saying something false, so the logic is inconsistent. Assuming, as is reasonable, that his logic (Russell and Whitehead's *Principia Mathematica*) is consistent, he concludes that the sentence is true and unprovable.

Gödel shows how to construct his true but unprovable sentence, but he does not construct it. It involves immense numbers, and the encoding is so opaque that we would not be enlightened by it. In order to examine Gödel's sentence, we shall use the transparent encoding of the previous sections: character strings.

Gödel defined the "is a theorem" predicate (he called it *Bew*) on numbers (a predicate is a function with a boolean range). We shall instead define $\vdash$ on strings so that $\vdash s$ is a theorem if and only if s represents a theorem. This may be done either by adding the symbol $\vdash$ to the formalism and giving the rules for its use, or by defining a function using the symbols already in the formalism. Either way, it amounts to writing a theorem-proving program. The important point is that we have a single theory serving both as the object of study and as the formalism in which to do the study.

Next, Gödel defined an embedding relation Q , again on numbers; we shall define Q as a function from strings to strings. When applied to a string that represents a predicate, it produces another string that represents a sentence by replacing all occurrences of substrings that represent free variables with the entire quoted string. For example, the string

$$\text{“}\exists u\cdot\ t = u\ u\text{”}$$

represents a predicate in free variable t . Applying Q to that string, we obtain the theorem

$$Q\ \text{“}\exists u\cdot\ t = u\ u\text{”}\ \ =\ \ \text{“}\exists u\cdot\ \underline{\text{“}}\exists u\cdot\ t = u\ u\underline{\text{”}}\ =\ u\ u\text{”}$$

The inner quotes are underlined to indicate that they are just characters in the string. (Every programming language has some special convention to allow quotes within strings.) Again, this function may be defined either by adding the symbol Q to the formalism and giving the rules for its use, or by defining a function using the symbols already in the formalism. Either way, it amounts to writing a program.

All the pieces are now in place. In particular, we have the theorem

$$Q\ \text{“}\neg\vdash Q\ s\text{”}\ \ =\ \ \text{“}\neg\vdash Q\ \underline{\text{“}}\neg\vdash Q\ s\underline{\text{”}}\ \text{”}$$

Applying $\neg\vdash$ to both sides of the above equation, we obtain the theorem

$$\neg\vdash Q\ \text{“}\neg\vdash Q\ s\text{”}\ \ =\ \neg\vdash\ \text{“}\neg\vdash Q\ \underline{\text{“}}\neg\vdash Q\ s\underline{\text{”}}\ \text{”}$$

Taking a liberty with substitution inside quotes, we can say that this theorem has the form

$$G = \neg\vdash \text{“}G\text{”}$$

with each occurrence of G standing for

$$\neg\vdash Q\ \text{“}\neg\vdash Q\ s\text{”}$$

This is the famous Gödel sentence but using a string encoding. If G were a theorem, then $\vdash \text{“}G\text{”}$ would be a theorem and $\neg\vdash \text{“}G\text{”}$ an antitheorem. Thus we would prove a theorem equal to an antitheorem, and the theory would be inconsistent. But we believe the theory is consistent. So we conclude that G is not a theorem. If G were an antitheorem, then $\neg\vdash \text{“}G\text{”}$ would be an antitheorem because they are proven equal, hence $\vdash \text{“}G\text{”}$ would be a theorem. Since $\vdash \text{“}G\text{”}$ is a theorem if and only if G is a theorem, G would be a theorem, which is inconsistent. Therefore G is not an antitheorem either. The theory is incomplete.

How shall we interpret G ? For it to represent a true statement, it must at least be a formal sentence. Is it? To establish that the Gödel utterance

$$\neg \vdash Q\,\text{“}\neg \vdash Q\,s\text{”}$$

is a sentence, we must show that

$$\vdash Q\,\text{“}\neg \vdash Q\,s\text{”}$$

is a sentence. If $\vdash$ applies only to strings that represent sentences, then we must show that

$$Q\,\text{“}\neg \vdash Q\,s\text{”}$$

represents a sentence. Since we can prove

$$Q\,\text{“}\neg \vdash Q\,s\text{”} = \text{“}\neg \vdash Q\,\underline{\text{“}}\neg \vdash Q\,s\underline{\text{”}}\,\text{”}$$

we can ask instead if the string

$$\text{“}\neg \vdash Q\,\underline{\text{“}}\neg \vdash Q\,s\underline{\text{”}}\,\text{”}$$

represents a sentence. It does if and only if

$$\neg \vdash Q\,\text{“}\neg \vdash Q\,s\text{”}$$

is a sentence, but that is the original question. The only way out is to allow that $Q\ s$ is a sentence for any string s , and/or that $\vdash s$ is a sentence for any string s , whether or not s represents a sentence. We must, at least temporarily, suspend our desire to interpret; we must consider only the formal manipulation of symbols in order to arrive at Gödel's result.

Given the reflexive axiom of equality, a formalist is willing to accept $0÷0 = 0÷0$ as a theorem of NT. He is not concerned with what mathematical object is referred to by $0÷0$. He is likewise willing to accept $\vdash$ “)+x=(” as a sentence of TT , though it is not a theorem. We defined $\vdash$ as a predicate on strings, so that $\vdash s$ is a sentence no matter what string s may be. Similarly Gödel, in his formal presentation, uses an encoding that assigns a distinct natural number to every sequence of symbols, whether it is an expression or not, and defines his predicate *Bew* on all natural numbers. Prior to the formal presentation of his result, Gödel gives an informal presentation for motivation, in which he proposes to encode only predicates onto the natural numbers: “We think of the class-signs [predicates] as being somehow arranged in a series, and denote the n-th one by $R(n)$”. This is possible because predicatehood is defined by a program; the program can print a list of all and only the predicates, assigning each the next natural number. This encoding has the attraction that metamathematical sentences and predicates can always be interpreted, through the code, as talking about sentences and predicates. But it is a fatal attraction. Indeed, in the introduction to the English translation of Gödel's paper, R.B.Braithwaite considered it a flaw of the formal presentation that Gödel encoded all sequences of symbols. Far from being a flaw, it was essential! Later we shall be in a position to make this point more simply and clearly.

Gödel's Second Incompleteness Theorem

Gödel's First Incompleteness Theorem says that a particular theory, if consistent, is incomplete. Its interest comes from the effort that was spent trying to make that theory complete. When a sentence is discovered that is neither a theorem nor an antitheorem, it can be made either one of those, at our choice, by adding an axiom. Gödel's Second Incompleteness Theorem says that this process of adding axioms can never make the theory complete (and still consistent).

When we add an axiom to a theory, we obtain a different theory. For the second theorem we must put back the theory name in front of ⊢ and ⊣ . The first theorem says that the sentence

¬ TT ⊢ *Q* “¬ TT ⊢ *Q s*”

is unclassified in theory TT . We can create theory TTT from theory TT by adding the previous sentence as an axiom. But the second theorem says that the sentence

¬ TTT ⊢ *Q* “¬ TTT ⊢ *Q s*”

will be unclassified. Speaking informally, the second theorem is the same as the first theorem but letting the theory name be variable. Henceforth we shall not distinguish between them.

Semantics and Interpreters

As we have seen, some sentences are theorems, some are antitheorems, and some are neither. It is tempting to say that the sentences which are neither are in a third class, and to invent a third symbol ⊥ to go with ⊢ and ⊣ . In some circles, three-valued logic is popular. Unfortunately, any attempt to formalize the third class will run into the same problem: there will be a gap, and a temptation to invent a fourth class, and so on. I prefer to say that those sentences that are neither theorem nor antitheorem are “unclassified”.

In this section we present a simpler and less expressive metatheory than in the previous sections, using only one symbol. For any theory T and string *s* we introduce the sentence

T 𝙸 *s*

Predicate 𝙸 is said to interpret string *s* in theory T . When it is clear which theory is meant, we may omit its name. For each theory, we want 𝙸 *s* to be a theorem if and only if *s* represents a theorem, and an antitheorem if and only if *s* represents an antitheorem. It is related to ⊢ and ⊣ by the two implications

⊢ *s* ⇒ 𝙸 *s* ⇒ ¬ ⊣ *s*

In fact, if we have defined ⊢ and ⊣ , those implications define 𝙸 . But we want 𝙸 to replace ⊢ and ⊣ so we shall instead define it by showing how it applies to every form of sentence. Here is the beginning of its definition.

𝙸 “*true*” = *true*
𝙸 “*false*” = *false*
∀*s*· 𝙸 (“¬” *s*) = ¬ 𝙸 *s*
∀*s*· ∀*t*· 𝙸 (*s* “∧” *t*) = 𝙸 *s* ∧ 𝙸 *t*
∀*s*· ∀*t*· 𝙸 (*s* “∨” *t*) = 𝙸 *s* ∨ 𝙸 *t*

And so on. Notice that 𝙸 acts as the inverse of quotation marks; it “unquotes” its operand. That is what an interpreter does: it turns passive data into active program. It is a familiar fact to programmers that we can write an interpreter for a language in that same language, and that is just what we are doing here.

To finish defining 𝙸 we must decide the details of an entire theory. We shall not do so here, but we give one more case of special interest. 𝙸 is defined on strings beginning with 𝙸 as

$\forall s\cdot\ \mathtt{I}\,(\text{“}\,\mathtt{I}\,\underline{\text{“}}\,\text{”}\ s\ \text{“}\,\underline{\text{”}}\,\text{”}) = \mathtt{I}\,s$

Thus the interpreter becomes part of the interpreted logic.

Gödel Simplified

Our presentation of Gödel's argument was parallel to his, but using strings instead of numbers. The argument would be essentially unchanged if we had used $\mathbb{I}$ in place of $\vdash$. The heart of the argument is a transformation from one level of quotes to two levels, and back down to one, with a "$\neg$" appearing in the process. We can simplify the argument without loss of content by going from zero to one and back. We don't take Q to be a function from strings to strings, but simply a string. It is defined as

$$Q = \text{“}\neg\ \mathbb{I}\ Q\text{”}$$

By its addition we do not make a theory incomplete: Q equals a completely known 3-character string. Now let us suppose that we can construct (or define) an unquoting function $\mathbb{I}$ (it does not matter how). Then we have the theorem

$\quad\ \mathbb{I}\ Q$
$= \mathbb{I}\ \text{“}\neg\ \mathbb{I}\ Q\text{”}$ replacing Q with its equal
$= \neg\ \mathbb{I}\ Q$ because $\mathbb{I}$ unquotes

and the inconsistency is apparent. So are the properties necessary to make the argument: a theory must allow us to replace something with its equal, and it must include or allow us to define its own interpreter. To save such a theory from inconsistency, we could suspend the ability to replace something with its equal under certain circumstances, but that is a distasteful option. Instead we leave the interpreter incomplete. In particular, if

$$\mathbb{I}\ \text{“}\neg\ \mathbb{I}\ Q\text{”} = \neg\ \mathbb{I}\ Q$$

is a theorem then we have inconsistency, and if it is an antitheorem then $\mathbb{I}$ is not an interpreter, so we leave it unclassified.

As before, we must consider $\mathbb{I}\ s$ to be a sentence for all strings s, even those that do not represent sentences. Otherwise we cannot show that $\mathbb{I}\ Q$ is a sentence. It is tempting to interpret Q as representing a sentence saying that it (Q) is not interpretable. But in a very real sense, we must refrain from interpreting Q: in programming terms, applying interpreter $\mathbb{I}$ to string Q will cause an infinite execution loop, and yield no result.

Conclusion

After the pioneering work of Frege, Russell, and other logicians at the beginning of this century, David Hilbert hoped it would be possible to formalize all of mathematics. But in 1931, Kurt Gödel arrived at a result which is commonly interpreted as saying that any formalism that includes arithmetic allows us to express truths of mathematics that cannot be proven in the formalism. And with that, Hilbert's hope died. The important point of Gödel's result is not the existence of true but unprovable sentences; it is easy to design an incomplete theory in which some of the unprovable sentences were intended to represent truths. Gödel's result says that no one formalism completely describes all formalisms (including itself). But it is equally true that every formalism is completely describable by another formalism, and in that more modest sense Hilbert's hope is achievable.

We believe that our presentation of Gödel's Incompleteness Theorem(s) nicely illustrates E.W.Dijkstra's contention that computing science can now repay with interest its debt to mathematics. Specifically, the distinction between program and data, the use of the character string data type, the use of an interpreter, and counting from zero, reduced the proof to three lines.

Acknowledgement. I thank Bill McKeeman for suggesting that I rescue this presentation of Gödel's theorem from a longer, philosophically inclined paper.

References

Gödel, K.: Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I, *Monatshefte für Mathematik und Physik* v.38 p.173-198, Leipzig, 1931.

Gödel, K.: *On Formally Undecidable Propositions Of Principia Mathematica And Related Systems*, translated into English by B. Meltzer, with Introduction by R.B. Braithwaite, Oliver&Boyd, Edinburgh, 1962.