Gödel's incompleteness theorem

Davide Testuggine

University of Cambridge

November 22nd, 2011

The author

Quick bio

- Austrian logician, though he spent most of his time in Princeton, USA
- Good friend of Einstein's
- Published the paper when he was 25, in 1931 (still in Austria)

Preliminar knowledge

Primitive recursive functions

A predicate or a function is **Primitive recursive** iff it is computable in a programming language which has only bounded loops (no while loops, no if + GOTO).

- This kind of functions is not Turing-complete
- Primitive recursive functions are always halting

Denotability and provability

- **Denotability** is the possibility to translate a predicate from the natural language to a formal system.
- **Provability** means that all true instances of a predicate are theorems of the system and all false ones are nontheorems.

Preliminar knowledge

Primitive recursive functions

A predicate or a function is **Primitive recursive** iff it is computable in a programming language which has only bounded loops (no while loops, no if + GOTO).

- This kind of functions is not Turing-complete
- Primitive recursive functions are always halting

Denotability and provability

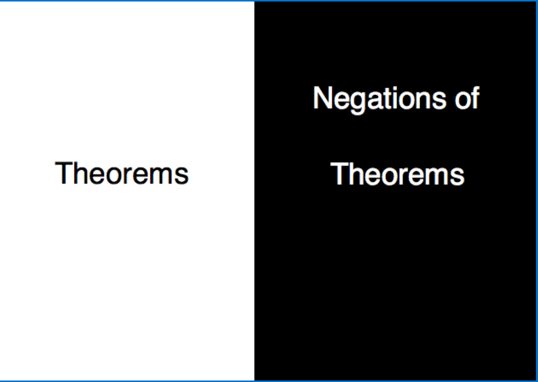
- **Denotability** is the possibility to translate a predicate from the natural language to a formal system.
- **Provability** means that all true instances of a predicate are theorems of the system and all false ones are nontheorems.

The first Incompleteness Theorem

Formulation

In every formal system able to prove primitive recursive statements, there will always be true predicates that cannot be proven (i.e. the system will always be incomplete)

What we would think

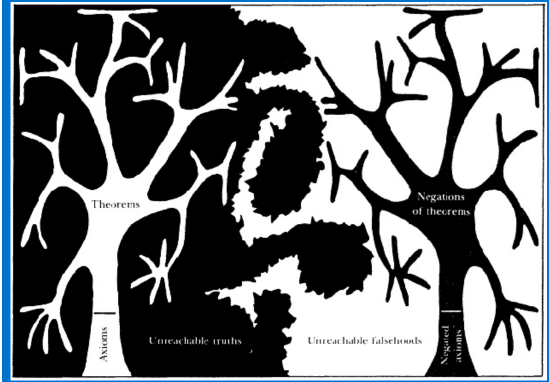


The first Incompleteness Theorem

Formulation

In every formal system able to prove primitive recursive statements, there will always be true predicates that cannot be proven (i.e. the system will always be incomplete)

What the reality is



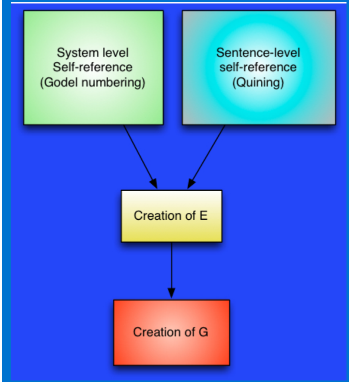
Proof sketch

The most important ideas

Two basic ideas:

1. Prove that the system of *Principia Mathematica* is capable of self-reference
2. Concentrate that self-reference in a **single sentence**
3. Prepare the ground by creating E, the "Egg" that will spawn the sentence
4. Evaluate G outside the system

Visual sketch



Therefore

Self-reference both at **system level** and at **sentence level**

Self-reference at system level

Challenge

We need to use Maths as English and make it speak about itself, such as "Predicate X is provable" just as in English we may say "Sentence X is grammatical"

Gödel's numbering

We translate every predicate and every symbol in a **number**.

A possible conversion table + Example

Symbol		*Codon*	*Mnemonic Justification*
0	.....	666	Number of the Beast for the Mysterious Zero
S	.....	123	successorship: 1, 2, 3, . . .
=	.....	111	visual resemblance, turned sideways
+	.....	112	1 + 1 = 2
·	.....	236	2 × 3 = 6
(	.....	362	ends in 2
)	.....	323	ends in 3
<	.....	212	ends in 2 — *these three pairs form a pattern*
>	.....	213	ends in 3
[	.....	312	ends in 2
]	.....	313	ends in 3
a	.....	262	opposite to ∀ (626)
′	.....	163	163 is prime
∧	.....	161	'∧' is a "graph" of the sequence 1-6-1
∨	.....	616	'∨' is a "graph" of the sequence 6-1-6
⊃	.....	633	6 "implies" 3 and 3, in some sense . . .
~	.....	223	2 + 2 is *not* 3
∃	.....	333	'∃' looks like '3'
∀	.....	626	opposite to a; also a "graph" of 6-2-6
:	.....	636	two dots, two sixes
punc.	.....	611	special number, as on Bell system (411, 911)

626,262,636,223,123,262,111,666
∀ a : ~ S a = 0

Self-reference at system level

Challenge

We need to use Maths as English and make it speak about itself, such as "Predicate X is provable" just as in English we may say "Sentence X is grammatical"

Gödel's numbering

We translate every predicate and every symbol in a **number**.

A possible conversion table + Example

Symbol		*Codon*	*Mnemonic Justification*
0	.....	666	Number of the Beast for the Mysterious Zero
S	.....	123	successorship: 1, 2, 3, ...
=	.....	111	visual resemblance, turned sideways
+	.....	112	1 + 1 = 2
·	.....	236	2 × 3 = 6
(	.....	362	ends in 2
)	.....	323	ends in 3
<	.....	212	ends in 2 — *these three pairs form a pattern*
>	.....	213	ends in 3
[	.....	312	ends in 2
]	.....	313	ends in 3
a	.....	262	opposite to ∀ (626)
′	.....	163	163 is prime
∧	.....	161	'∧' is a "graph" of the sequence 1-6-1
∨	.....	616	'∨' is a "graph" of the sequence 6-1-6
⊃	.....	633	6 "implies" 3 and 3, in some sense ...
~	.....	223	2 + 2 is *not* 3
∃	.....	333	'∃' looks like '3'
∀	.....	626	opposite to a; also a "graph" of 6-2-6
:	.....	636	two dots, two sixes
punc.	.....	611	special number, as on Bell system (411, 911)

626,262,636,223,123,262,111,666
∀ a : ~ S a = 0

Self-reference at system level

Challenge

We need to use Maths as English and make it speak about itself, such as "Predicate X is provable" just as in English we may say "Sentence X is grammatical"

Gödel's numbering

We translate every predicate and every symbol in a **number**.

A possible conversion table + Example

Symbol		*Codon*	*Mnemonic Justification*
0	.....	666	Number of the Beast for the Mysterious Zero
S	.....	123	successorship: 1, 2, 3, . . .
=	.....	111	visual resemblance, turned sideways
+	.....	112	1 + 1 = 2
·	.....	236	2 × 3 = 6
(	.....	362	ends in 2
)	.....	323	ends in 3
<	.....	212	ends in 2 — *these three pairs form a pattern*
>	.....	213	ends in 3
[	.....	312	ends in 2
]	.....	313	ends in 3
a	.....	262	opposite to ∀ (626)
′	.....	163	163 is prime
∧	.....	161	'∧' is a "graph" of the sequence 1-6-1
∨	.....	616	'∨' is a "graph" of the sequence 6-1-6
⊃	.....	633	6 "implies" 3 and 3, in some sense . . .
~	.....	223	2 + 2 is *not* 3
∃	.....	333	'∃' looks like '3'
∀	.....	626	opposite to a; also a "graph" of 6-2-6
:	.....	636	two dots, two sixes
punc.	.....	611	special number, as on Bell system (411, 911)

626,262,636,223,123,262,111,666
∀ a : ~ S a = 0

Why Gödel's numbering is the solution

What we have done

Every proof is now a single, **huge** number

⇒ Relations between theorems and their proofs are now only numerical!

Can we really do this?

To do this, our arithmetical system still needs to be able to **evaluate** the predicate *proofFor(proof, theorem)*.

As that predicate is **primitive recursive**, this is what we ask to our system. The system in *Principia Mathematica* is strong enough to do it (we are not proving this, and neither did Gödel. I think it is inside PM itself)

Why Gödel's numbering is the solution

What we have done

Every proof is now a single, **huge** number
$\Rightarrow$ Relations between theorems and their proofs are now only numerical!

Can we really do this?

To do this, our arithmetical system still needs to be able to **evaluate** the predicate *proofFor(proof, theorem)*.

As that predicate is **primitive recursive**, this is what we ask to our system. The system in *Principia Mathematica* is strong enough to do it (we are not proving this, and neither did Gödel. I think it is inside PM itself)

Why Gödel's numbering is the solution

What we have done

Every proof is now a single, **huge** number
⇒ Relations between theorems and their proofs are now only numerical!

Can we really do this?

To do this, our arithmetical system still needs to be able to **evaluate** the predicate *proofFor(proof, theorem)*.

As that predicate is **primitive recursive**, this is what we ask to our system. The system in *Principia Mathematica* is strong enough to do it (we are not proving this, and neither did Gödel. I think it is inside PM itself)

Why Gödel's numbering is the solution

What we have done

Every proof is now a single, **huge** number
$\Rightarrow$ Relations between theorems and their proofs are now only numerical!

Can we really do this?

To do this, our arithmetical system still needs to be able to **evaluate** the predicate *proofFor(proof, theorem)*.
As that predicate is **primitive recursive**, this is what we ask to our system. The system in *Principia Mathematica* is strong enough to do it (we are not proving this, and neither did Gödel. I think it is inside PM itself)

Further considerations

Another predicate

Let us consider *provable(formula)*. This is not primitive recursive, but PM may still **denote** it by saying $\exists$ *x.proofFor(x, formula)* (and translating into a big Gödel number).

Conclusion for this

We have found a way to express a statement **on** the system within the system itself. We cannot resolve it directly, but if we find some other way to do it, then we will be fine!

Further considerations

Another predicate

Let us consider *provable(formula)*. This is not primitive recursive, but PM may still **denote** it by saying $\exists$ *x.proofFor(x, formula)* (and translating into a big Gödel number).

Conclusion for this

We have found a way to express a statement **on** the system within the system itself. We cannot resolve it directly, but if we find some other way to do it, then we will be fine!

Self-reference at sentence level

Challenge

To talk about ourselves, we may say "Me" in English.
How can we do the same in Maths?

Idea

We may talk about ourselves in terms of **description**.
We will use **Quination** for this.

Self-reference at sentence level

Challenge

To talk about ourselves, we may say "Me" in English.
How can we do the same in Maths?

Idea

We may talk about ourselves in terms of **description**.
We will use **Quination** for this.

Self-reference at sentence level

Challenge

To talk about ourselves, we may say "Me" in English.
How can we do the same in Maths?

Idea

We may talk about ourselves in terms of **description**.
We will use **Quination** for this.

Quination

Definition

To apply a sentence to a quotation of itself.

Example

"Has got four words" has got four words

Effect

If you consider a sentence as a predicate, you are using itself as its own argument, thus creating a loop.

What happens

This sentence talks about itself now! Basically, it is the same as stating "I have four words"!

Quination

Definition

To apply a sentence to a quotation of itself.

Example

"Has got four words" has got four words

Effect

If you consider a sentence as a predicate, you are using itself as its own argument, thus creating a loop.

What happens

This sentence talks about itself now! Basically, it is the same as stating "I have four words"!

Quination

Definition

To apply a sentence to a quotation of itself.

Example

"Has got four words" has got four words

Effect

If you consider a sentence as a predicate, you are using itself as its own argument, thus creating a loop.

What happens

This sentence talks about itself now! Basically, it is the same as stating "I have four words"!

Quination

Definition

To apply a sentence to a quotation of itself.

Example

"Has got four words" has got four words

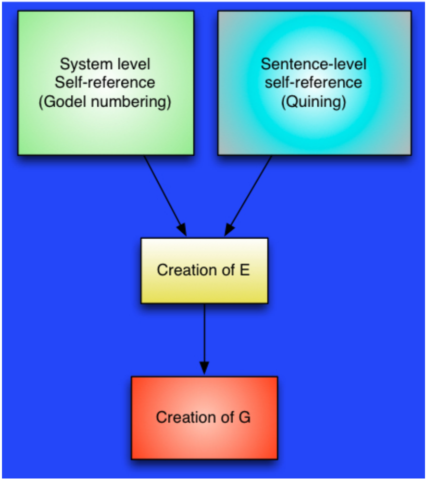
Effect

If you consider a sentence as a predicate, you are using itself as its own argument, thus creating a loop.

What happens

This sentence talks about itself now! Basically, it is the same as stating "I have four words"!

Diagram


System level
Self-reference
(Godel numbering)
Sentence-level
self-reference
(Quining)
Creation of E
Creation of G

Putting things together

Our aim

We want to express the following sentence:
"I am not a theorem of PM"

Preparing to translate

We quination to talk about ourselves. As first, we prepare ourselves defining E, the Egg of G:
E := $\neg\exists Proof.\exists$ Theorem. [proofFor(Proof, Theorem) $\wedge quination(x, Theorem)$]

Translation

Now, the final shot: we quine E itself, so we make E talk about itself.
G := $\neg\exists Proof.\exists$ Theorem. [proofFor(Proof, Theorem) $\wedge quination(E, Theorem)$]

Putting things together

Our aim

We want to express the following sentence:
"I am not a theorem of PM"

Preparing to translate

We quination to talk about ourselves. As first, we prepare ourselves defining E, the Egg of G:
E := $\neg\exists Proof.\exists$ Theorem. [proofFor(Proof, Theorem) $\wedge quination(x, Theorem)$]

Translation

Now, the final shot: we quine E itself, so we make E talk about itself.
G := $\neg\exists Proof.\exists$ Theorem. [proofFor(Proof, Theorem) $\wedge quination(E, Theorem)$]

Evaluation

If we feed the Gödel number of G to the system, then Resolution **can't** evaluate this, but **we** are outside the system and can do its in one shot:

False case

if G was **false**, then it would be a theorem of PM. **But you cannot derive false predicates from your axioms!**

True case

if G was **true**, then it would not be provable - but it is not a contradiction, so we must accept this.

Q.E.D

Evaluation

If we feed the Gödel number of G to the system, then Resolution **can't** evaluate this, but **we** are outside the system and can do its in one shot:

False case

if G was **false**, then it would be a theorem of PM. **But you cannot derive false predicates from your axioms!**

True case

if G was **true**, then it would not be provable - but it is not a contradiction, so we must accept this.

Q.E.D

Evaluation

If we feed the Gödel number of G to the system, then Resolution **can't** evaluate this, but **we** are outside the system and can do its in one shot:

False case

if G was **false**, then it would be a theorem of PM. **But you cannot derive false predicates from your axioms!**

True case

if G was **true**, then it would not be provable - but it is not a contradiction, so we must accept this.

Q.E.D

Final thoughts

1. If you add G to the axioms, there will always be a G' and so on (there is also "ω - incompleteness").
2. John Lucas, a Oxford philosopher, argued in his article **Minds, machines and Gödel** that this theorem proves that Church-Turing thesis is false, i.e. that machines can't compute everything that a human can.
3. **Easy to misunderstand**. Real and Complex numbers theory and Geometry theory are **not** formal systems and in fact they are **complete** (see Tarski's proof).
4. There is another theorem in the original paper: it states that *if your system is coherent, then it cannot prove its own coherence*.
5. The second Incompleteness theorem solved the famous *Entscheidungsproblem* from Hilbert (saying "No"). This is the same conclusion of the famous paper of Turing introducing the machine and Alonzo Church's paper on λ calculus.

Thank you!

Thank you for your time! :-)

Questions?

If you are curious about something, or if I wasn't clear in some passages ask me and I will try my best!

Thank you!

Thank you for your time! :-)

Questions?

If you are curious about something, or if I wasn't clear in some passages ask me and I will try my best!