Gödel's Theorem

1 The Theorem

Let $\mathcal{L}$ be a (rich enough) arithmetical language:

Gödel's Incompleteness Theorem (V1) No Turing Machine can do the following: when given a sentence of $\mathcal{L}$ as input, it outputs "1" if the sentence is true and "0" if the sentence is false.

Gödel's Incompleteness Theorem (V2) No Turing Machine can:

1. run forever, outputting sentences of $\mathcal{L}$;
2. eventually output each true sentence of $\mathcal{L}$; and
3. never output a false sentence of $\mathcal{L}$.

Gödel's Incompleteness Theorem (V3) No axiomatization of $\mathcal{L}$ is both consistent and complete.

2 What the Theorem Teaches Us

- *Mathematically:*

  Arithmetical truth is too complex to be finitely specifiable.

- *Philosophically:*

  Interesting mathematical theories can never be established beyond any possibility of doubt.

3 A Simple Arithmetical Language, L

- An **arithmetical language** is a language to talk about the natural numbers and their basic operations (e.g. addition and multiplication).

- L is an arithmetical language built from the following symbols:[1]

[1] With some effort, the exponentiation symbol can be defined using "+" and "×", as Gödel showed. I include it here because it will make proving the theorem much easier.

Arithmetical Symbol	Denotes
0	the number zero
1	the number one
$+$	addition
$\times$	multiplication
$\wedge$	exponentiation

Logical Symbol	Read
$=$	. . . is identical to . . .
$\neg$	it is not the case that . . .
$\&$	it is both the case that . . . and . . .
$\forall$	every number is such that . . .
x_n (for $n \in \mathbb{N}$)	it

Auxiliary Symbol	Meaning
(	[left parenthesis]
)	[right parenthesis]

3.1 Abbreviations

An **abbreviation** is a notational shortcut to make it easier for us to keep track of certain strings of symbols on our official list.

- *Numerals:*

Abbreviation	Read	Official Notation
2	two	$(1 + 1)$
3	three	$((1 + 1) + 1)$
4	four	$(((1 + 1) + 1) + 1)$
$\vdots$	$\vdots$	$\vdots$

- *Logical Symbols:*

Abbreviation	Read	Official Notation
$A \vee B$	A or B	$\neg(\neg A \ \& \ \neg B)$
$A \supset B$	if A, then B	$\neg A \vee B$
$\exists x_i$	some number is such that it	$\neg \forall x_i \neg$

- *Arithmetical Symbols:*

Abbreviation	Read	Official Notation
$x_i < x_j$	x_i is smaller than x_j	$\exists x_k((x_j = x_i + x_k) \ \& \ \neg(x_k = 0))$
$x_i \vert x_j$	x_i divides x_j	$\exists x_k(x_k \times x_i = x_j)$
Prime(x_i)	x_i is prime	$(1 < x_i) \ \& \ \forall x_j \forall x_k((x_i = x_j \times x_k) \supset (x_i = x_j \vee x_i = x_k))$

4 A Rich Enough Language

$\mathcal{L}$ counts as "rich enough" if one can prove:

Lemma $\mathcal{L}$ contains a formula (abbreviated "Halt(k)"), which is true if and only if the kth Turing Machine halts on input k.

(As it turns out, even our simple language L satisfies this condition!)

5 A Proof of Gödel's Theorem (V1)

- Assume for *reductio*: M decides the truth of sentences of $\mathcal{L}$.
- By the Lemma, we can use M's program to construct a Turing Machine M^H, which computes the Halting Function.
- Since the Halting Function is not Turing-computable, our assumption must be false.

We've proved Gödel's Thoerem!