A Simple Character String Proof of the "True but Unprovable" Version of Gödel's First Incompleteness Theorem

Antti Valmari

Tampere University of Technology

FINLAND

1 Introduction

Gödel's tremendously influential *first incompleteness theorem* 1931

- Gödel's version (what it means is explained later)

  Each recursive sufficiently strong theory of natural numbers either is ω-inconsistent or leaves both a sentence and its negation without a proof.

- modern (computer science) version

  Each recursively enumerable sufficiently expressive theory of natural numbers either proves a false sentence or fails to prove a true sentence.

- popularized version

  There are mathematical truths that cannot be proven.
  This occurs already with elementary school mathematical concepts.

- Gödel's version avoids mentioning *truth* as separate from provability

The details of Gödel's proof are hard to understand

- difficult and verbose concepts in the precise formulation
- *self-reference* — making a sentence talk about itself
- *Gödel numbering* — encoding reasoning in arithmetic

This paper: simplify, by *proving the modern version first for finite character strings*

2 Language of Arithmetic

The following is sufficiently expressive

- constants 0 and 1
- binary arithmetic operations $+$ and $\cdot$
- logical operations $\neg$, $\wedge$, and $\forall$
- parentheses $($ and $)$
- unbounded supply of natural number variable symbols x, y, ...

Additional notation can be defined as abbreviations

- $4 \ := \ (1+1+1+1)$
- $x \leq y \ :\Leftrightarrow \ (\exists z : y = x + z)$
- $\varphi \vee \psi \ :\Leftrightarrow \ \neg(\neg\varphi \vee \neg\psi)$
- $\mathsf{Prime}(x) \ :\Leftrightarrow \ (x \geq 2 \wedge \neg\exists y : \exists z : y \geq 2 \wedge z \geq 2 \wedge x = y \cdot z)$

Can you define x^y ?

- this fails, because it is circular: $\mathsf{Pow}(x, y, z) \ :\Leftrightarrow$
  $$y = 0 \wedge z = 1 \ \vee \ \exists y' : \exists z' : y = y' + 1 \wedge z = x \cdot z' \wedge \mathsf{Pow}(x, y', z')$$
- it can be defined, but it is not easy
- $\cdot$ cannot be defined from 0, 1, $+$, $\neg$, ...

3 “Reasonable” Reasoning

Unrealistic proof systems must be ruled out

- example: every true sentence is an axiom, there are no reasoning rules
  - every true sentence has a proof
  - proofs are conveniently short
  - problem: we do not know which formulae are axioms

Modern assumption

Proof system must be recursively enumerable.

- there must be a computer program that inputs a finite character string and
  - eventually halts, if the input string is a valid proof
  - never halts, otherwise
- computer program $:=$ Turing machine
- in essence, proofs must be “one-way” checkable
- this suffices for the modern version
- Gödel assumed strictly more (but also gave a stronger consequent):
  - there must be a computer program that eventually replies “yes” or “no”
  - proof system must be sufficiently strong, e.g., Peano’s axioms & first-order

4 Main Idea of Gödel's Proof 1/2

Gödel encoded sentences and proofs as natural numbers

- $\mathsf{Prf}(p, s)$ $:\Leftrightarrow$ p is the number of a proof of the sentence, whose number is s
- made before programming was familiar $\Rightarrow$ major achievement
- the proof system is so strong that
  - if $\mathsf{Prf}(p, s)$ holds, then it can be proven
  - if $\mathsf{Prf}(p, s)$ does not hold, then $\neg\mathsf{Prf}(p, s)$ can be proven

$\Rightarrow$ proofs are "both-ways" checkable, proof system is recursive

Gödel found a way to make a formula "know" its Gödel number

- let
  - $\varphi(x)$ be a formula (possibly) with free variable x, and f its Gödel number
  - $\mathbf{n}$ be a natural number
- a formula σ exists such that $\sigma(f, \mathbf{n}, f')$ $\Leftrightarrow$ f' is the Gödel number of $\varphi(\mathbf{n})$
- let $\psi(x)$ be any formula with (at most) the free variable x
- let $\psi'(x)$ $:\Leftrightarrow$ $\exists y : \psi(y) \wedge \sigma(x, x, y)$ and let $\mathbf{g}$ be its Gödel number

$\Rightarrow$ $\psi'(\mathbf{g})$ is equivalent to $\psi(\mathbf{y})$, where $\mathbf{y}$ is the Gödel number of $\psi'(\mathbf{g})$

- that is, $\psi'(\mathbf{g})$ says *"I am a sentence that has the property ψ."*

4 Main Idea of Gödel's Proof 2/2

These allowed Gödel write a sentence G such that

- G is provably equivalent to $\neg\exists x : \mathsf{Prf}(x, \mathbf{g})$, where $\mathbf{g}$ is the Gödel number of G

$\Rightarrow$ G says *"I am a sentence that has no proof."*

Assume that G has a proof

- then also $\neg\exists x : \mathsf{Prf}(x, \mathbf{g})$ has a proof, because G is provably equivalent to it
- let $\mathbf{p}$ be the Gödel number of a proof of G
  $\Rightarrow$ the system proves $\mathsf{Prf}(\mathbf{p}, \mathbf{g})$ and $\exists x : \mathsf{Prf}(x, \mathbf{g})$

$\Rightarrow$ the proof system proves a contradiction and is *inconsistent*

Assume that G does not have and $\neg G$ has a proof

- none of $\mathsf{Prf}(0, \mathbf{g})$, $\mathsf{Prf}(1, \mathbf{g})$, $\mathsf{Prf}(2, \mathbf{g})$, $\ldots$ holds
  $\Rightarrow$ the system proves $\neg\mathsf{Prf}(0, \mathbf{g})$, $\neg\mathsf{Prf}(1, \mathbf{g})$, $\neg\mathsf{Prf}(2, \mathbf{g})$, $\ldots$
- the system proves $\exists x : \mathsf{Prf}(x, \mathbf{g})$

$\Rightarrow$ the proof system is *ω-inconsistent*
  - proves that a number with a certain property exists, but disallows all values

If neither G nor $\neg G$ has a proof, then the proof system is *incomplete*

5 Main Idea of Our Proof 1/2

Strategy

- prove the modern version for finite character strings with concatenation
- carry the result to the language of arithmetic

Finite character strings with concatenation

- *characters*

  `a b c d e f g h i j k l m n o p q r s t u v w x y z`
  `0 1 2 3 4 5 6 7 8 9 " \ = ≠ ( ) ~ & | - > A E : + * <`

- *string literals* `backslash="\"`
  - examples: `"hello!"`, `"backslash=\1\0\1"`, `""`
  - `\` and `"` are *encoded* by `\0` and `\1`, other chars are encoded by themselves
- *variables*: ⟨lower case letter⟩ ⟨digit⟩*
- *terms*: (⟨string literal⟩ | ⟨variable⟩)$^+$
- *atomic propositions*: ⟨term⟩`=`⟨term⟩ | ⟨term⟩`≠`⟨term⟩
- *formulae*: ⟨atomic prop.⟩ | `(`⟨form.⟩`)` | `~`⟨form.⟩ | ⟨form.⟩`&`⟨form.⟩ | ⟨form.⟩`|`⟨form.⟩ | ⟨form.⟩`->`⟨form.⟩ | `A`⟨variable⟩`:`⟨form.⟩ | `E`⟨variable⟩`:`⟨form.⟩
- *sentence* := formula with no free variables

5 Main Idea of Our Proof 2/2

Gödel's self-referential sentence *"I am a sentence that has no proof"* is now

```
 Ex:Ey: Q(x, y) & ~Pvble(x"\1"y"\1") & x=
"Ex:Ey: Q̇(x, y) & ~Ṗvble(x"\1"y"\1") & x="
```

where

- Q(x, y) (shown later) says that y contains the encoding of x
- Pvble(x) (shown later) says that x has a proof
- Q̇(x, y) is Q(x, y) with each \ and " replaced by \0 and \1, and Ṗvble(x) . . .

⇒ x"\1"y"\1" is the sentence

The end of the proof is immediate:

- if the sentence has a proof, it is a false sentence with a proof
- if the sentence lacks the proof, it is a true sentence without proof

We still have to

- construct Q(x, y)
- construct Pvble(x)
- carry the result to natural numbers

6 Encoding of Strings 1/2

Replacement of one substring by another is easy to express:

```
RepOne(x, u, v, y) :⇔ ( Ee:Ef: x=euf & y=evf )
```

Replacement of every instance of some substring requires trickery

- represented as a sequence of replacements
  - a punctuation substring p separates the steps of the sequence
  - to avoid picking fake instances of u, some assumptions are now made and later enforced
- here it is:

```
RepAll(x, u, v, y, p) :⇔
(Es:(Et:s=pxpt)&(Et:s=tpyp)&~(Ez:Ev:y=zuv)&Ah:Ak:((Eu:Ev:s=
uphpkpv)&~(Eu:Ev:h=upv)&~(Eu:Ev:k=upv))->(Ee:Ef:h=euf&k=evf))
```

- perhaps some abbreviations and lay-out tricks help to grasp it

```
RepAll(x, u, v, y, p) :⇔ ( Es:
        (Et: s=pxpt) & (Et: s=tpyp) & ~Sb(u, y)
      & Ah:Ak: ( Sb(phpkp, s) & ~Sb(p, h) & ~Sb(p, k) )
                -> RepOne(h, u, v, k)
)
```

6 Encoding of Strings 2/2

`+:::` is used for punctuation and `*:::` is used for intermediate results

- not necessarily three `:` are used, but sufficiently many
  - must avoid fake instances of `+:::` and `*:::`
  - $|x| + 1$ certainly suffices

```
Punct(x, q) :⇔ ((Aa: Sb(a, q) & Char(a) -> a=":") & ~Sb(q, x) )
```

- `*:::` and `+:::` cannot overlap

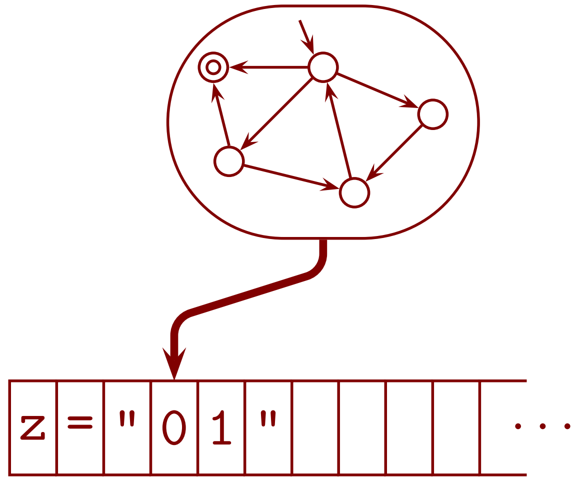
Here it is:

```
Q(x, y)  :⇔  ( Eq: Punct(x, q)
      & Ex1: RepAll( x, "\0",  "*"q, x1, "+"q)           \   ⇝  *:::
      & Ex2: RepAll(x1, "*"q, "\00", x2, "+"q)        *:::   ⇝  \0
      & Ex3: RepAll(x2, "\1",  "*"q, x3, "+"q)           "   ⇝  *:::
      &      RepAll(x3, "*"q, "\01",  y, "+"q)        *:::   ⇝  \1
)
```

- `y` is obtained from `x` by replacing every `\` by `\0` and every `"` by `\1`

This completes our version of self-reference

7 Simulating Turing Machines



The proof system is recursively enumerable

$\Rightarrow$ there is a Turing machine T that halts
if and only if its input is a theorem

$\Rightarrow$ Pvble(x) $\Leftrightarrow$ T halts on x

A *configuration* of T consists of

- contents of the tape strictly left of the read/write head
  - represented backwards and without unnecessary blanks at left end
- state of the finite control, represented as a number (sequence of digits)
- contents of the tape under and to the right of the read/write head
  - represented without unnecessary blanks at right end

Pvble(x) says that there is c such that

- c consists of encoded configurations separated with suitable punctuation
- the first configuration is the initial configuration of T with input x
- the last configuration is a halted configuration of T
- each configuration is obtained from the previous by some rule of T

8 Carrying the Result to Arithmetic 1/2

A one-to-one correspondence between strings and natural numbers

- let
  - $p = 53$
  - the characters be numbered 1, 2, $\ldots$, 53
  - c_i' denote a character and c_i denote its number
  - ι_n be the string of n characters with number 1: $\iota_n = \overbrace{\texttt{aaa}\cdots\texttt{a}}^{n}$
- the string $c_1'c_2'\cdots c_n'$ has the number

$$num(c_1'c_2'\cdots c_n') \;=\; c_1p^{n-1} + c_2p^{n-2} + \ldots + c_{n-2}p^2 + c_{n-1}p + c_n$$

- the proof that this mapping is 1–1 is straightforward (and not new)

We show soon that there is a formula γ in the language of arithmetic such that $\gamma(num(\sigma), num(\rho), num(\zeta))$ holds if and only if the concatenation of σ and ρ is ζ

$\Rightarrow$ each sentence on strings can be automatically translated to arithmetic

$\Rightarrow$ any r.e. proof system for arithmetic is also a r.e. proof system for strings
  - if it proves all and only true sentences on arithmetic,
    it proves all and only true sentences on strings $\lightning$

$\Rightarrow$ *Each recursively enumerable proof system for* 0, 1, $+$, $\cdot$, $=$, $\neg$, $\wedge$, $\forall$, $($, *and* $)$ *either proves a false sentence or fails to prove a true sentence.*

8 Carrying the Result to Arithmetic 2/2

Mapping concatenation to arithmetic

$$\underbrace{num(c'_1 \cdots c'_n d'_1 \cdots d'_m)}_{\mathtt{z}} = \underbrace{p^m}_{\mathtt{k}} \underbrace{num(c'_1 \cdots c'_n)}_{\mathtt{x}} + \underbrace{num(d'_1 \cdots d'_m)}_{\mathtt{y}}$$

- let
  - $\mathtt{x} = num(c'_1 \cdots c'_n)$
  - $\mathtt{y} = num(d'_1 \cdots d'_m)$
  - $\mathtt{z} = num(c'_1 \cdots c'_n d'_1 \cdots d'_m)$

$\Rightarrow$ $num(\iota_m) \leq \mathtt{y} < num(\iota_{m+1})$

$\Leftrightarrow$ $\sum_{i=0}^{m-1} p^i \leq \mathtt{y} < \sum_{i=0}^{m} p^i$

$\Leftrightarrow$ $p^m - 1 \leq (p-1)\mathtt{y} < p^{m+1} - 1$

$\Leftrightarrow$ $p^m + \mathtt{y} \leq \mathtt{y}p + 1 < pp^m + \mathtt{y}$

$\Rightarrow$ the following formula expresses concatenation

```
Ek: Pow53(k) & y*53+1 < 53*k+y & ~(y*53+1 < k+y) & z=k*x+y
```

where

```
Pow53(k) :⇔ ( Ax:Ay: k=x*y -> x=1 | Ez: x=53*z )
```

- Pow53(k) works because 53 is a prime number

This completes our version of Gödel numbers

9 Rosser's Formulation

Rosser [1936] replaced inconsistency for ω-inconsistency in Gödel's version

> *Each recursive sufficiently strong theory of natural numbers either is inconsistent or leaves both a sentence and its negation without a proof.*

His self-referential sentence R is equivalent to

$$\forall x : (\ \mathsf{Prf}(x, \mathbf{r}) \rightarrow \exists y : y \leq x \wedge \mathsf{Prf}(y, \bar{\mathbf{r}})\ )$$

- $\mathbf{r}$ is the Gödel number of R
- $\bar{\mathbf{r}}$ is the Gödel number of $\neg R$

Assume that $\mathbf{p}$ is the Gödel number of a proof of R

- the system proves R, $\mathsf{Prf}(\mathbf{p}, \mathbf{r})$, and $\exists y : y \leq \mathbf{p} \wedge \mathsf{Prf}(y, \bar{\mathbf{r}})$
- if such an y indeed exists, then it proves $\neg R$ and the system is inconsistent
- otherwise the system proves $\neg\exists y : y \leq \mathbf{p} \wedge \mathsf{Prf}(y, \bar{\mathbf{r}})$ via proving $\neg\mathsf{Prf}(0, \bar{\mathbf{r}}) \wedge \cdots \wedge \neg\mathsf{Prf}(\mathbf{p}, \bar{\mathbf{r}})$ and the system is inconsistent

Assume that $\mathbf{p}$ is the Gödel number of a proof of $\neg R$

- the system proves $\mathsf{Prf}(\mathbf{p}, \bar{\mathbf{r}})$ and $\exists x : \mathsf{Prf}(x, \mathbf{r}) \wedge \neg(\mathbf{p} \leq x)$

$\Rightarrow$ if x exists, then $R \wedge \neg R$, otherwise $\neg\mathsf{Prf}(0, \mathbf{r}) \wedge \cdots \wedge \neg\mathsf{Prf}(\mathbf{p} - 1, \mathbf{r})$

10 Discussion

Today Gödel's and Rosser's formulations are somewhat forgotten, because

- Gödel's formulation needs stronger and more verbose assumptions
- today "true but unprovable" is not considered problematic
  - halting of Turing machines gives a solid notion of truth

Self-reference can be made simpler still

- modern proof of non-existence of halting testers is very simple
  $\Rightarrow$ negation of halting is not recursively enumerable
- provability is recursively enumerable by definition

$\Rightarrow$ incompleteness is obtained, if halting is automatically represented as a formula

- this approach yields simpler self-reference but more complicated Gödel numbering and simulation of Turing machines than this paper

Incompleteness of strings is too obvious to be new

The idea of proving it directly, then carrying the result to arithmetic seems new

Thank you for attention!