# LSS 2025: Computability and Incompleteness

## IV. Gödel's First Incompleteness Theorem, Tarski's Indefinability of Truth

Michael Norrish
michael.norrish@anu.edu.au

Australian National University

# Outline



# Last Time. . .

**First Order Logic**:

- Sound & complete, with computable rules of inference.
- Thus: recursively enumerable (semi-decidable).
- Expressive enough to capture behaviour of a Turing Machine.
- Thus: undecidable.

**Gödel Numbering**:

- Can convert formulas into numbers
- Can (computably) perform formula operations within arithmetic

**The Logic $\mathcal{Q}$**

- A basis for incompleteness results to come.

# Tying the Knot

After some technical details, today we will see that:

- Arithmetic is too powerful, too expressive. . .
- Too powerful for any logical system to capture entirely

- Essentially, because arithmetic is powerful enough to diagonalise
  - to pull Cantor's trick on the logical system

- Note though that we come to this realisation through the careful manipulations of a logical system!

# Outline



# Function Representability

If our logic is to have any oomph at all, it needs to be able to reason about functions other than addition and multiplication.

Say that a function $f : \mathbb{N}^n \to \mathbb{N}$ is representable in theory $\mathcal{T}$ if there is a formula $A(x_1, \ldots, x_n, x_{n+1})$ such that

- if $f(x_1, \ldots, x_n) = y$, then

  $$\vdash_{\mathcal{T}} \forall x_{n+1}.\, A(\mathbf{x_1}, \ldots, \mathbf{x_n}, x_{n+1}) \iff (x_{n+1} = \mathbf{y})$$

  where $\mathbf{x_i}, \mathbf{y}$ are translations of the numbers $x_i$ and $y$ into terms of the logic.
  - *E.g.*, in $\mathcal{Q}$, $\mathbf{3} = s(s(s(\mathbf{0})))$

For example, the function $f_=(x, y)$ that returns $1$ if $x = y$ and $0$ otherwise is representable in $\mathcal{Q}$ by

$$(x = y \wedge r = s(\mathbf{0})) \vee (x \neq y \wedge r = \mathbf{0})$$

# Recursive Functions are Representable in $\mathcal{Q}$

We will show that all recursive (computable) functions are representable in $\mathcal{Q}$. (No mean feat!)

## Base Cases:

- zero function (input $x$): $(y = \mathbf{0})$
- successor function (input $x$): $(y = s(x))$
- projection $p_{i,n}$: $(y = x_i)$

Remember it's necessary to show equivalence is derivable in $\mathcal{Q}$ (which is weak).

What's trivial in all arithmetic may not be so easy in $\mathcal{Q}$.

# Composition is Representable in $\mathcal{Q}$

Want to represent: $\mathsf{Cn}[f, g_1, \ldots, g_n]$

Assume $f$, $g_i$ are representable by $F(x_1, \ldots, x_n, x)$ and $G_i(z_1, \ldots, z_m, y)$ respectively.

Representation of composition is a formula with $z_1$ to $z_m$ (inputs) and $x$ (output) free.

$$\exists y_1 \ldots y_n.$$
$$G_1(z_1, \ldots, z_m, y_1) \wedge \cdots \wedge G_n(z_1, \ldots, z_m, y_n) \wedge F(y_1, \ldots, y_n, x)$$

# Minimisation is Representable in $\mathcal{Q}$

Must first have notion of $<$.

Define $x < y \stackrel{\text{def}}{=} \exists z.\ s(z) + x = y$

- Everywhere I write $x < y$, that is a short-hand for the RHS above; I haven't extended $\mathcal{Q}$.

Assume $f : \mathbb{N}^{n+1} \to \mathbb{N}$ is representable by $F$.

Want to show $\mathsf{Mn}[f] : \mathbb{N}^n \to \mathbb{N}$ is also representable.

With $x_1 \ldots x_n$ as inputs, and $y$ as output:

use $F(y, x_1, \ldots, x_n, \mathbf{0}) \wedge (\forall i.\ i < y \ \Rightarrow\ \neg F(i, x_1, \ldots, x_n, \mathbf{0}))$

# Representability of Primitive Recursion

Complicated! [See *B&J* for details on how this needs to be structured.]

We want to emulate $\mathsf{Pr}[f, g]$ (base-case $f$, recursive-case $g$)

Such that

$$\begin{aligned} \mathsf{Pr}[f,g](x_1,\ldots,x_n,0) &= f(x_1,\ldots,x_n) \\ \mathsf{Pr}[f,g](x_1,\ldots,x_n,m+1) &= g(x_1,\ldots,x_n,m,\mathsf{Pr}[f,g](x_1,\ldots,x_n,m)) \end{aligned}$$

Basic idea is to construct a list of all the answers to the recursive calls (up to some limit $k$).

- Recall argument about TM emulation of primitive recursion

# Representability of Primitive Recursion

Constructing a list of answers. . .

Write $R(x_1, \ldots, x_n, k, \ell)$ as abbreviation for

$$\begin{array}{l} \mathsf{index}(\ell, \mathbf{0}) = f(x_1, \ldots, x_n) \wedge \\ \forall j < k.\ \mathsf{index}(\ell, s(j)) = g(x_1, \ldots, x_n, j, \mathsf{index}(\ell, j)) \end{array}$$

Read $R(\vec{x}, k, \ell)$ as "$\ell$ represents the values of $\mathsf{Pr}[f, g]$ up to $k$"

Can assume that $f$ and $g$ are representable.

- Will need to demonstrate that we can encode lists,
- and the index function

# Give Me Your Huddled Answers Up to $k$

With $R$ to hand, can represent

$$d(x_1, \ldots, x_n, k) = \mu\,\ell.\ R(x_1, \ldots, x_n, k, \ell)$$

(using $\mu\, x \ldots$ for "least $x$" operator)

The $d$ function is representable if index, $f$ and $g$ are.

---

Then $\mathsf{Pr}[f, g] = \mathsf{Cn}[\mathsf{index}, d, p_{n+1,n+1}]$, and so representable.

$$\begin{aligned}
\mathsf{Pr}[f, g](x_1, \ldots, x_n, k) &= \mathsf{Cn}[\mathsf{index}, d, p_{n+1,n+1}](x_1, \ldots, x_n, k) \\
&= \mathsf{index}(d(x_1, \ldots, x_n, k), p_{n+1,n+1}(x_1, \ldots, x_n, k)) \\
&= \mathsf{index}(d(x_1, \ldots, x_n, k), k)
\end{aligned}$$

# Encoding Lists and the Indexing Operation

Approach from *B&J* (there are surely others).

To represent $e_1, \ldots, e_k$

1. Pick a prime $p$ such that $p - 1 > k$ and $p - 1 > e_i$ for all $i$
2. Let $s = p - 1$
3. For element $e_i$ use "cell": $s\, i\, e_i$ (digits concatenated together).
4. For whole list, concatenate all cells in order
5. Treat digit sequence as number to base $p$
6. Pair resulting number with $p$

For example, represent $[2, 1, 3, 4]$ with $p = 7$ as

$$\langle 7, 602611623634_7 \rangle \qquad (= \langle 7, 11980304662_{10} \rangle)$$

# Lists, Digit Lists Concatenated, Primes, Oh My!

Just a sample of some of the technology required:

Concatenation:

Write $a *_p b$ to mean $a$ concatenated with $b$ (using base $p$).

Definition: $a *_p b = a \cdot \eta(p, b) + b$

The $\eta$ function calculates how far left a number needs to be shifted to make room for $b$:

$$\eta(p, b) = \mu i.\ (p \text{ is prime} \wedge i \text{ is a power of } p \wedge i > b \wedge i > 1) \vee (p \text{ is not prime} \wedge i = 0)$$

# Representability in Summary

All computable functions can be represented in first order logics with nothing more than addition and multiplication.

We earlier saw that FOL could capture whether or not Turing Machines would halt.

Quantifiers buy you a lot!

# Outline



# Getting Ready to Diagonalise

Assume we have a Gödel-numbering function $gn : formula \to \mathbb{N}$.

Remember we also have a representation for individual $\mathbb{N}$ inside the logic.

- $\mathbf{3} = s(s(s(\mathbf{0})))$
- Generally, $\mathbf{n} = s^n(\mathbf{0})$

This is a function from $\mathbb{N}$ to terms (call it $nt$).

Compose the two ($nt \circ gn$), and we get a function from formulas to terms, mapping $A$ to $\ulcorner A \urcorner$.

- $\ulcorner A \urcorner$ is the "Quine-quote" of $A$

# Diagonalising a Formula

Call

$$\exists x.\ (x = \ulcorner A \urcorner) \wedge A$$

the diagonalisation of $A$.

Note: if $x$ is free in $A$, then the diagonalisation of $A$ is a formula that says

$A$ *is true of "itself".*

or, more accurately,

$A$ *is true of its own Gödel-number.*

# Computable Diagonalisation

Given $n = gn(A)$, it is easy to compute $gn(\text{diagonalisation of } A)$.

- Recall diagonalisation of $A$ is $\exists x.\ (x = \ulcorner A \urcorner) \wedge A$

A definition: $[diag : \mathbb{N} \to \mathbb{N}]$

$$diag(n) = \text{mk\_exists}(Vx, \text{mk\_conj}(\text{mk\_eq}(Vx, nt(n)), n))$$

where $Vx = \text{mk\_var}(\texttt{"x"})$, and
where the $\text{mk\_}$ functions compute Gödel-numbers
of the right form, given Gödel-numbers as inputs.

Properties:

$$\begin{aligned} diag(gn(A)) &= gn(\exists x.\ x = \ulcorner A \urcorner \wedge A) \\ gn^{-1}(diag(n)) &= \exists x.\ x = nt(n) \wedge gn^{-1}(n) \end{aligned}$$

# The Diagonal Lemma

Let $\mathcal{T}$ be a theory in which $diag$ is representable. Then for any formula $B(y)$ (of the language of $\mathcal{T}$, containing just the variable $y$ free), there is a sentence $G$ such that

$$\vdash_{\mathcal{T}} G \iff B(\ulcorner G \urcorner)$$

Let $DG$ be the predicate that represents $diag$. Then, for all $n$

$$\vdash_{\mathcal{T}} \forall y.\, DG(nt(n), y) \iff y = nt(diag(n))$$

Let $F$ be $\exists y.\, DG(x, y) \wedge B(y)$, and $G$ the diagonalisation of $F$.



# Proving the Diagonal Lemma

$DG$ represents $diag$: for all $n$, $\vdash_{\mathcal{T}} \forall y.\ DG(nt(n), y) \iff y = nt(diag(n))$
$F = \exists y.\ DG(x, y) \wedge B(y)$, $G$ is diagonalisation of $F$.

$$
\begin{array}{rcl}
G & = & \exists x.\ (x = \ulcorner F \urcorner) \wedge F \\
& \iff & \exists x\, y.\ (x = \ulcorner F \urcorner) \wedge DG(x, y) \wedge B(y) \\
& \iff & \exists y.\ DG(\ulcorner F \urcorner, y) \wedge B(y) \\
& = & \exists y.\ DG(nt(gn(F)), y) \wedge B(y) \\
& \iff & \exists y.\ (y = nt(diag(gn(F)))) \wedge B(y) \\
& = & \exists y.\ (y = nt(gn(G))) \wedge B(y) \\
& \iff & B(\ulcorner G \urcorner)
\end{array}
$$

# More Vocabulary—Theories

A theory is a set of theorems.

$\mathcal{T}_2$ is an extension of $\mathcal{T}_1$ if $\mathcal{T}_1 \subseteq \mathcal{T}_2$.

A theory $\mathcal{T}$ is

| | |
|---|---|
| Consistent | if it does not contain both $A$ and $\neg A$ for any $A$. |
| Complete | if, for all (closed) $A$, it contains either $A$ or $\neg A$. |
| Decidable | if there is a computable function that correctly characterises Gödel-numbers of theorems. |
| Axiomatisable | if there is a decidable subset of $\mathcal{T}$ whose logical consequences are all of $\mathcal{T}$. |

Note: “Complete” is overloaded; inference systems can be complete too.

# More Vocabulary—Definability

A set of natural numbers $S$ is definable in $\mathcal{T}$ if there is a formula $B(n)$ such that

$$n \in S \quad \text{iff} \quad \vdash_{\mathcal{T}} B(\mathbf{n})$$
$$n \notin S \quad \text{iff} \quad \vdash_{\mathcal{T}} \neg B(\mathbf{n})$$

Recursive/decidable sets are definable in extensions of $\mathcal{Q}$.

Analogously, can talk of $n$-place relations being definable.

# Indefinability Lemma 1

If $\mathcal{T}$ is a consistent extension of $\mathcal{Q}$, then the set of Gödel-numbers of theorems of $\mathcal{T}$ is not definable.

By contradiction.

- Let $Thm$ be the formula capturing theorem-hood.
  - So, $\vdash_{\mathcal{T}} Thm(nt(n))$ iff $\vdash_{\mathcal{T}} gn^{-1}(n)$ (1)
    and $\vdash_{\mathcal{T}} \neg Thm(nt(n))$ iff $\nvdash_{\mathcal{T}} gn^{-1}(n)$ (2)
- The function $diag$ is representable in $\mathcal{T}$.
- So, there is a $G$, such that $\vdash_{\mathcal{T}} G \iff \neg Thm(\ulcorner G \urcorner)$ (†)
- Is $G$ a theorem of $\mathcal{T}$?
  - Yes. Then $\vdash_{\mathcal{T}} Thm(\ulcorner G \urcorner)$ by (1). But then, by (†) $\vdash_{\mathcal{T}} \neg G$ also!
    ($\mathcal{T}$ inconsistent, a contradiction)
  - No. Then $\vdash_{\mathcal{T}} \neg Thm(\ulcorner G \urcorner)$ by (2). So, by (†), $\vdash_{\mathcal{T}} G$.
    ($G$ a theorem after all, contradiction)

# Undecidability of Arithmetic

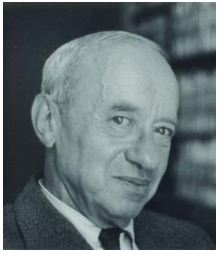
Arithmetic is the set of true statements about $\mathbb{N}$.

- Thus, a consistent extension of $\mathcal{Q}$.

If arithmetic were decidable, it would be definable.

But we know that theorem-hood is not definable in consistent extensions of $\mathcal{Q}$.

# Tarski's Indefinability Theorem



Alfred Tarski
(1901–1983)

The set of Gödel-numbers of sentences true in arithmetic is not definable in arithmetic.

Immediate from Indefinability Lemma 1 as arithmetic is a consistent extension of $\mathcal{Q}$.

When Gödel and Tarski were both living in the USA (at Princeton and Berkeley respectively), Tarski is said to have described himself as

*the greatest sane logician in the USA.*

# An Axiomatisable, Complete Theory is Decidable

Uses fact that inference system for FOL is complete (in “sound/complete” sense).

Assume that $S$ is the decidable set of axioms.

Algorithm to decide if $A$ is a theorem:

1. Enumerate all theorems.
2. Eventually, one of $S_0 \Rightarrow A$ or $S_0 \Rightarrow \neg A$ will appear (by completeness—other sense!)
   - $S_0$ is a finite conjunction of elements of $S$

# Gödel's First Incompleteness Theorem

Kurt Gödel
(1906–1978)

There is no consistent, complete, axiomatisable extension of $\mathcal{Q}$.

Corollary: arithmetic is not axiomatisable.

- You can pick any two of consistent, complete and axiomatisable.
- And we assume that the Platonic theory of arithmetic is the first two.

Gödel starved himself to death in the belief that he was being poisoned.

# Why Gödel's *Incompleteness* Theorem?

> *There is no consistent, complete, axiomatisable extension of* $\mathcal{Q}$*.*

A formal system (axioms + rules of inference) is axiomatisable as long as the axioms are decidable.

So, when we apply to Gödel's First Incompleteness Theorem to formal systems

- (that include a modest amount of arithmetic)

It tells us that if it is consistent, it must be incomplete.

Or, if it is complete, it is inconsistent (= useless).

# Summary

## Representability

- All recursive functions are representable in extensions of $\mathcal{Q}$

## Arithmetic Cannot be Captured

- The diagonalisation function is computable
- So any candidate “theorem-hood” notion can be turned against itself
  - “I am true iff I am not a theorem”
- Truth is not definable in arithmetic (Tarski)
- Arithmetic is not axiomatisable (Gödel)