Lecture 19: Computability

MATH230

Te Kura Pāngarau | School of Mathematics and Statistics
Te Whare Wānanga o Waitaha | University of Canterbury

Outline

Entscheidungsproblem

In 1928 David Hilbert asked if there could be an “effective procedure” that could decide (Yes/No) whether a sentence of first-order logic is a theorem.

Furthermore, he asked if there could be such a procedure that could determine whether a sentence is a theorem of some first-order theory of mathematics.

With the formal language written down, we now turn to the idea of an “effective procedure” or “algorithm”. In order to say whether such a thing exists, we must be precise about what we mean.

Mechanized Computation

- Euclid's Algorithm (Calculate greatest common divisor)
- Antikythera Mechanism (Ancient Greece, astronomical cycles)
- Difference Engine (Babbage, Built 1822)
- Analytical Engine (Babbage, Designed 1837)
- Mathematicians formalise the idea of computation $\sim$1900s.
- Z3 (Zuse, Built 1941)

In this course we're going to consider some early attemtps at formalising the idea of computation. What might it mean, in some generality, to say a process is computable.

The Computer: from Pascal to von Neumann. Herman Goldstine.

Gödel/ASCII/Unicode

We introduced Gödel numbering as a way of counting the wff of a first-order theory with countably many variables and a finite signature. However, Gödel originally introduced the idea in his proof of the incompleteness theorems.

Gödel did more than just show that there is *some* sentence for which it (and it's negation) can't be proved. He gave an effective procedure for generating such a sentence that one could get their hands on.

Gödel numbering was his way to make wff into objects of computation.

Rather than manipulating strings with some rules of computation, Gödel turned the problem into manipulating numbers with some rules of computation.

Numbers

We will focus on computations with numbers. Most commonly (in mathematics) we use the decimal system (powers of 10) to represent numbers. However, numbers can be represented as sums of powers of different bases.

For example

- (Decimal) $123_{10} = 1 \times 10^2 + 2 \times 10^1 + 3 \times 10^0$
- (Binary) $123_{10} = 1111011_2$
  $1 \times 2^6 + 1 \times 2^5 + 1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$
- (Unary) $123_{10} = 11111 \cdots 1111_1$
  $1 \times 1^{123} + 1 \times 1^{122} + \cdots + 1 \times 1^1 + 1 \times 1^0$

Computation?

What should we mean by an “effective procedure”? What properties do we think such a procedure should have?

Gödel's Proof

A lot of the work in the proof of the incompleteness theorem relied on finding effective definitions of predicates for the following:

- Does x divide y?
- Is ϕ (the Gödel number of) an axiom?
- Is ϕ the negation of ψ?
- Is there an effective function to generate $\phi \wedge \psi$?
- Does ϕ follow from ψ by modus ponens?

In his paper Gödel provides recursive definitions of relations and functions culminating in the definition of the following

- Is ϕ the Gödel number of a proof?
- ϕ is a proof of ψ.

By effective procedure Gödel meant primitive recursion.

Primitive Recursion: Initial Functions

Primitive recursive functions $f : \mathbb{N}^k \to \mathbb{N}$ are built from the following basic functions.

Zero: $Z(n) = 0$
Successor: $Succ(n) = s(n)$
Projections: $\pi_i^k(x_1, \ldots, x_i, \ldots x_k) = x_i$

These are chosen as there is no questioning the computability of these functions. In all cases it's clear what the output should be.

More sophisticated primitive recursive functions are built inductively from these initial functions according to finitely many applications of function composition and recursion.

Function Composition

If $g : \mathbb{N}^m \to \mathbb{N}$ is primitive recursive and $h_1, \ldots, h_m : \mathbb{N}^k \to \mathbb{N}$ are each primitive recursive, then the function $f : \mathbb{N}^k \to \mathbb{N}$ defined by function composition

$$f(\mathsf{x}) = g(h_1(\mathsf{x}), \ldots, h_m(\mathsf{x}))$$

is a primitive recursive function.

Example

Recursion: Single Variable

If g is a primitive recursive function and $d \in \mathbb{N}$, then the function $f : \mathbb{N} \to \mathbb{N}$ defined by

$$f(0) = d$$
$$f(s(n)) = g(f(n), n)$$

is also primitive recursive.

Example

Multiplication and Exponentiation

Provide recursive definitions of multiplication and exponentiation.

Recursion: Multiple Variable

If g, h are primitive recursive functions of multiple variables, then the following function

$$f(0, \mathrm{x}) = g(\mathrm{x})$$
$$f(s(n), \mathrm{x}) = h(f(n, \mathrm{x}), n, \mathrm{x})$$

is also primitive recursive.

Effective Procedure as Recursion

One interpretation of what it means for a procedure to be effective, is if there exists a primitive recursive function that computes the output.

It's up to us to combine the initial components by composition and recursion to see how powerful this family of functions is.

Predecessor

Show that the predecessor function is primitive recursive.

Limited Subtraction (Monus)

Show that limited subtraction is primitive recursive.

Zero Test

The zero test is primitive recursive.

Signature

The non-zero test, or signature function, is primitive recursive.

Absolute Difference

The absolute difference is primitive recursive.

Min, Max

Min (or max) of two variables is primitive recursive.

Further Reading

Here are some recommended reading to follow up on the lecture content.

- SEP: Recursive Functions.
- Computability, Richard Epstein.
- Computability Theory, Enderton.
- Lectures on the Philosophy of Mathematics, Joel Hamkins.