Arithmetic and Incompleteness

Will Gunther

February 6, 2013

About Talk

Things talk

- Will approach from angle of computation.
- Will not assume very much knowledge.
- Will "prove" Gödel's Incompleteness Theorem.
- Will not talk much about first order logic.
- Will not even write down any axioms of arithmetic.
- Will not talk about every detail.

Things to Take Away

1. Arithmetic is powerful.
2. Incompleteness is an obvious corollary of (1).
3. Incompleteness is not frustrating.

The big theorems

There are three “big theorems” which make up incompleteness. We will prove two.

- **Gödel’s β Function Lemma** There is a very computable way to code sequences of natural numbers.
- **Gödel’s Representability Theorem** All primitive recursive functions can be represented in Peano’s Arithmetic (omitted).
- **Gödel’s Diagonal Lemma** Formulas have “fixed points”

What are Natural Numbers?

- The Natural Numbers are the numbers $0, 1, 2, \ldots$.
- We can define them inductively as the smallest set containing 0, and closed under the operation of taking a successor.
- This is a circular definition in the eyes of mathematical foundations.

Problem to Ponder: How can we better define the natural numbers to be more pure with respect to foundations? This question invites writing down axioms for how numbers behave.

What can we do with Natural Numbers?

- We will be particularly diligent in deciding what we can do with natural numbers. For instance, we will not give ourselves the power to do arbitrary calculations on the natural numbers.
- Instead, we want to capture what simple operations we can do on natural numbers. There are several approaches.
- Approach One: Addition and multiplication are the only thing we can do.
  Result: Arithmetic is fairly boring.
- Approach Two: We can do addition, multiplication, and define things by induction.
  Result: Arithmetic becomes self-aware.

Primitive Recursion

A function $f : \mathbb{N}^n \to \mathbb{N}$ is primitive recursive if and only if it is one of the following:

- $f(x_1, \ldots, x_n) = 0$
- $f(x_1, \ldots, x_n) = s(x_1)$ where s is the successor operation.
- $f(x_1, \ldots, x_n) = x_i$ for some $1 \leq i \leq n$.
- $f(x_1, \ldots, x_n) = h(g_1(x_1, \ldots, x_n), \ldots, g_k(x_1, \ldots, x_n))$ where h, g primitive recursive.
- $f(x_1, \ldots, x_n) = \begin{cases} g(x_1, \ldots, x_n) & \text{if } x_1 = 0 \\ h(x_1, \ldots, x_n, f(x^*, x_2, \ldots, x_n)) & \text{if } x_1 = s(x^*) \end{cases}$
  where h, g are primitive recursive.

What is Primitive Recursive

- What is a function that is not primitive recursive?
  Answer: It doesn't matter.
- In a computability class, primitive recursive functions are just the first stopping point.
- For us, it's all(ish) we need. Because...

Fact

Most functions are primitive recursive.

Coding with Primitive Recursive Functions

We have the above language of primitive recursive functions, and our goal is the following theorem:

Theorem (Gödel's β function lemma)

There is a primitive recursive function $\beta : \mathbb{N}^2 \to \mathbb{N}$ such that for any sequence of natural numbers $\langle a_1, a_2, \ldots, a_n \rangle$ there is a natural number a such that for every $1 \leq i \leq n$

$$\beta(a, i) = a_i$$

a is called the code for the sequence $\langle a_1, \ldots, a_n \rangle$

The above theorem is the heart of incompleteness. It should tell you, if you look at the naturals just as $0, 1, 2, \ldots$ then you're wrong. The information that is encoded in the natural numbers is immense.

Let's add and multiply first...

As a toy project, let's define the function $+ : \mathbb{N}^2 \to \mathbb{N}$ which represents addition. This is a simple definition by recursion:

$$x + y := \begin{cases} \pi_2(x, y) & \text{if } x = 0 \\ s(x^* + y) & \text{if } x = s(x^*) \end{cases}$$

Now, it's not difficult to define multiplication.

$$x \cdot y := \begin{cases} 0 & \text{if } x = 0 \\ y + (x^* \cdot y) & \text{if } x = s(x^*) \end{cases}$$

Now let's subtract

Subtracting is a little more tricky perhaps. Note it's not always possible. For instance, what is $5 - 10$? So we restrict ourselves to cut-off subtraction. That is, subtraction but it cuts off at 0. First, we define the predecessor function, which is not too hard.

$$p(x) := \begin{cases} 0 & \text{if } x = 0 \\ x^* & \text{if } x = s(x^*) \end{cases}$$

Now, doing $x - y$ is just a matter of iterating this operation several times!

$$x - y := \begin{cases} x & \text{if } y = 0 \\ p(x - y^*) & \text{if } y = s(y^*) \end{cases}$$

Coding Booleans

For our purposes, $\top$ will be the constant function 1, and $\bot$ will be the constant function 0. Now, we define some simple booleans operations.

- $x \wedge y := x \cdot y$
- $x \vee y := (x + y) - (x \cdot y)$
- $\neg x := 1 - x$

Cases

We will define a function $x?y : z$ which outputs y if x is $\top$ and z if x is $\bot$ as follows:

$$(x?y : z) := \begin{cases} z & \text{if } x = 0 \\ y & \end{cases}$$

Relations and Characteristic Functions

A binary relation on $\mathbb{N}$ can be expressed as a function

$$f(x, y) = \begin{cases} \top & \text{if } R(x, y) \\ \bot & \end{cases}$$

Using this, we can talk about defining a relation using primitive recursive functions too. The relation $\leq$ is definable.

$$x \leq y := (x - y)?\bot : \top$$

Then of course equality and $<$ can be defined:

$$x = y := (x \leq y) \wedge (y \leq x)$$
$$x < y := (x \leq y) \wedge \neg(x = y)$$

Bounded Search

I can return the first value of x smaller then b for which some relation is true.

$$\mu_{x<b} f(x) := \begin{cases} 0 & \text{if } b = 0 \\ \begin{array}{r} ((\mu_{x<b^*} f(x)) = b^*)? \\ (f(b^*)?b^* : b) : (\mu_{x<b^*} f(x)) \end{array} & \text{if } b = s(b^*) \end{cases}$$

This easily allow us to do to ask if there is some $x < b$ such that some function is true.

$$\exists_{x<b} f(x) := ((\mu_{x<b} f(x)) = b)?\bot : \top$$

And one can write $\forall_{x<b} f(x) := \neg(\exists_{x<b} \neg f(x))$

Back to Division

Now, we can determine whether x divides y.

$$x \mid y := \exists_{z<y} x \cdot z = y$$

This also gives us a primality test.

$$\text{isPrime}(x) := \forall_{z<x} (z = 1) \vee \neg(z \mid x)$$

And we can even calculate the nth prime with the knowledge there is a prime between p and $2p$.

$$\text{pr}(n) := \begin{cases} 2 & \text{if } n = 0 \\ \mu_{z<2\cdot\text{pr}(n^*)} (z > \text{pr}(n^*)) \wedge \text{isPrime}(z) & \end{cases}$$

Integer Division and Modulus

We can calculate an integer division.

$$x \div y := y - (\mu_{z<y} x \cdot (y - z) \leq y)$$

And the remainder is of course:

$$x \% y := \mu_{z<y} (y \cdot (x \div y) + z) = y$$

Theorem (Gödel's β function lemma)

There is a primitive recursive function $\beta : \mathbb{N}^2 \to \mathbb{N}$ such that for any sequence of natural numbers $\langle a_1, a_2, \ldots, a_n \rangle$ there is a natural number a such that for every $1 \leq i \leq n$

$$\beta(a, i) = a_i$$

a is called the code for the sequence $\langle a_1, \ldots, a_n \rangle$

Proof of Gödel's β

Proof.

Step 1: We find a way to encode a pair $\langle a, b \rangle$. There are a few ways to do this. The earliest example is due to Cantor, and is the "dovetailing" bijection you probably have seen. Another technique is with Kleene's Pairing Function:

$$\pi(a, b) = 2^a(2b + 1)$$

We want to know that we can decode this using a primative recursive function.

$$\pi_1(p) = \mu_{z<p}((p \div 2^z)\%2 = 1)$$
$$\pi_2(p) = ((p \div 2^{\pi_1(p)}) - 1) \div 2$$

□

Chinese Remainder Theorem

Recall the following theorem from antiquity.

Theorem (Chinese Remainder Theorem)

For every sequence $a_1, \ldots, a_n$, if $p_1, \ldots, p_n$ are relatively prime then there is a number u such that

$$\begin{aligned} u &\equiv a_1 \mod p_1 \\ u &\equiv a_2 \mod p_2 \\ &\vdots \\ u &\equiv a_n \mod p_n \end{aligned}$$

u is the unique such number less than $\prod p_i$

β lemma proof continued

β lemma proof continued...

Step 2: Be clever, and use CRT. Consider the sequence $\langle a_1, \ldots, a_n \rangle$. Let N be the maximum of $a_1, \ldots, a_n, n$.
Claim that $N! + 1$, $2N! + 1$, $\ldots nN! + 1$ are all relatively prime.
Otherwise, there is some j that divides two of them, so it divides the difference, so it divides $N!$, so $j < N$. But of course no $j < N$ can divide $kN! + 1$.
Let u be obtained by CRT so that $u \equiv a_i \mod iN! + 1$.
Code the sequence $\langle a_1, \ldots, a_n \rangle$ as the pair $\pi(N!, u)$.

$$\beta(U, i) = \pi_2(U) \,\%\, (i \cdot \pi_1(U) + 1)$$

□

To Logic

- We have avoided talking about formal logic thus far, and we will continue to avoid a lot of details.
- The important thing is, using the β function, we can represent all the information we'd ever want to about logic in arithmetic.
  - $\ulcorner x_i \urcorner := \langle 0, i \rangle$
  - $\ulcorner \phi \wedge \psi \urcorner := \langle 1, \ulcorner \phi \urcorner, \ulcorner \psi \urcorner \rangle$
  - $\ulcorner \forall x.\phi \urcorner := \langle 2, \ulcorner x \urcorner, \ulcorner \phi \urcorner \rangle$
  - etc.

  These are call Gödel numbers of the formulas. Every formula has a Gödel number. Now, questions about logic can be answers just by arithmetic of the numbers.

Theorem

There is a primitive recursive function isWFF which can identify if a given natural number is the Gödel number of a formula.

What's in a Proof?

- A proof is a sequence of formulas where each is either an axiom or obtained from previous formulas by modus ponens (ie. if P and $P \to Q$ are listed earlier, we can now list Q).
- As formulas can be Gödel numbered with natural numbers, proofs can also be Gödel numbered as they are nothing more than sequences of formulas.
- We would like it if there were a function which recognizes whether a Gödel number is a valid proof.
- This might not always be the case for every axiomatic system. What is required is that the axioms are "simple" to describe.

Assumption: Simple list of Axioms

- Our system is something in the language of arithmetic (so there is $+$ and $\cdot$ and 0 and 1)
- We will assume that the axioms of our system are simple enough there there is a primitive recursive function that can decide whether a given formula is an axiom (so there is a primitive recursive function that can decide if a sequence of formulas is a proof).
- This is a reasonable assumption. (Peano's Arithmetic and ZFC both have simple axiom system, for example).

Assumption: Expressive

We assume our system is sufficiently expressive. That is, the following is true :
For every primitive recursive function $f(x_1, \ldots, x_n)$ there is a formula $\phi(x_1, \ldots, x_n, y)$ such that

$$f(x_1, \ldots, x_n) = y \iff \vdash \phi(x_1, \ldots, x_n, y)$$

This was proven by Gödel to hold for Peano Arithmetic. We will not prove this.

Fixed Point Theorem

Theorem

For every formula $\phi(x)$ with one free variable, there is a sentence ψ such that

$$\vdash \psi \leftrightarrow \phi(\ulcorner \psi \urcorner)$$

Assume this is true momentarily.

Incompleteness

Theorem

Our system is incomplete.

Proof.

We need to find a sentence ψ such that neither ψ nor $\neg\psi$ have a proof.

- Let $\phi(x)$ be the formula $\exists y. y$ is the Gödel number of a proof of $\neg x$.
- By the fixed point theorem these is ψ such that $\phi(\ulcorner\psi\urcorner) \leftrightarrow \psi$.
- Thus ψ is true if and only if there is a proof of $\neg\psi$.
- As we are assuming our system doesn't prove contradictions, we can neither prove ψ nor $\neg\psi$

$\square$

Proof of Fixed Point Theorem

Proof.

Step 1: The function **App** : $\mathbb{N}^2 \to \mathbb{N}$ is primitive recursive, which does the following:

$$\mathbf{App}(n, m) = \ulcorner \phi(m) \urcorner$$

Where $\ulcorner \phi(x) \urcorner = n$. This isn't hard to see; you just do cases on what kind of formula n represents and do the substitution inductively.

Define $f : \mathbb{N} \to \mathbb{N}$ by

$$f(x) = \mathbf{App}(x, x)$$

□

Proof of Fixed Point Theorem continued

Proof.

Step 2: Recall our language is expressive. So there is some formula $\theta_f(x, y)$ such that:

$$\theta_f(x, y) \iff y = f(x)$$

Consider the formula:

$$\mu(x) := \forall y \,.\, \theta_f(x, y) \to \phi(y)$$

It is easy to see that this formula is equivalent to $\phi(f(x))$; therefore we have:

$$\mu(x) \iff \phi(\mathbf{App}(x, x))$$

□

Proof of Fixed Point Theorem continued

Proof.

Step 3: Instantiate the formula $\mu(x)$ at it's own Gödel number, $\ulcorner \mu(x) \urcorner$. Then:

$$\begin{aligned} \mu(\ulcorner \mu(x) \urcorner) &\iff \phi(\mathbf{App}(\ulcorner \mu(x) \urcorner, \ulcorner \mu(x) \urcorner)) \\ &\iff \phi(\ulcorner \mu(\ulcorner \mu(x) \urcorner) \urcorner) \end{aligned}$$

So, set $\psi := \mu(\ulcorner \mu(x) \urcorner)$. So $\psi \iff \phi(\ulcorner \psi \urcorner)$. $\square$