Proving Towards Gödel's First Incompleteness Theorem

Jerry Yan

Contents

Preamble

This text is geared towards anyone interested in logic, but who doesn't have a very rich background in math and/or philosophy. The text involves symbolic logic and induction. At UofT, this would equate to finishing *MAT137 (Calculus I w/ Proofs)*, or *CSC165 (Math and Logic for CS)*. Lastly, to anyone who stumbled upon this text and willing to give it a read, thank you for your interest!

A little on the background, in recent years (especially this year), the field of computer science has seen tremendous progress with research in artificial intelligence, specifically machine learning techniques that could make machines mimic real humans, one prominent example being the famous ChatGPT.

However, such programs often lack the ability to reason logically. It is challenging for a chatbot to recognize their mistakes as they speak. This brings the notion of "axiomatized systems", which is studied in a sub-field of classical AI called knowledge representation; in this field, computers are used to formulate logical proofs based on given axioms. Just imagine a world where we have such a machine that constantly pumps out new knowledge. How cool is that!

Similar ambition dates back to as early as the 1920s, when David Hilbert proposed that it would only be a matter of time before we find all axioms of math, such that all maths could be derived from these axioms. This vision was exciting, but it was eventually shown to be impossible by Gödel's incompleteness theorem. This insight similarly applies to the idea of logical AI, as in, there is inhenerent limitations to what an axiomatized inference engine can do. In essence, such a machine can never be all-knowing.

Chapter 1

Preliminaries

The entirety of this text is dedicated to understanding **Gödel's First Incompleteness Theorem**, as a preview, it roughly states:

Every consistent formal system capable of doing elementary arithmetic must be incomplete.

One way to picture a formal system is by thinking of it as an "expert system" (a concept in classical AI); it is equipped with a knowledge-base KB, containing say, all physics knowledge of the world, and it can answer questions by proving or disproving them.

Naturally, the system's knowledge base KB first starts with a set of axioms, and thanks to the inference engine (inference rules) equipped by the expert system, it can populate its KB by adding new inferred theorems into it. If X is a physical phenomena explainable by our existing body of physics knowledge, then undoubtedly, X must be a part of KB.

1.1 Formal Theory

Definition 1.1.1. ***Formal Theory (Abstract Definition)***
A formal theory is a set of axioms equipped with rules of inference, such that theorems can be derived and also placed into the set. Formally, it consists of the following components:

1. *A language $\mathcal{L}$ composed of an alphabet and a grammar.*

- *The alphabet is a finite set of primitive symbols; we can concatenate these symbols to form strings called formulas.*
- *A grammar (syntactic rules) builds up formulas from simpler formulas. The formulas which satisfy our grammar are called well-formed formulas or wffs.*

2. *A set of axioms consisting of wffs.*
3. *A set of inference rules to infer theorems from axioms.*

As an example for alphabet and grammar, consider an alphabet $\{0, 1, (,), +\}$ with the grammar "every left parenthesis is closed by a right parenthesis", and "+ symbol is sandwiched by 2 numbers". In this case, "$(0 + ($" is not grammatical, hence not a *wff*, while "$(1 + 0)$" is a *wff*.

Moving onwards, $\mathcal{L}$ as a language has its *syntax* and *semantics*:

- **Syntax**: Syntax deals with the rules to construct and transform logical statements. In particular, it determines which strings of symbols form a "term" (a mathematical object that serves as a component of a formula), which strings form "*wff*s", and which strings form "sentences" (boolean-valued *wff*s with no free, unassigned variables).
- **Semantics**: Semantics give interpretation to sentences. For each sentence according to our syntax, we assign it to be True, or False.

Definition 1.1.2. ***Syntactic & Semantic Consequences***

Let Σ be a set of sentences, we say:

- *Sentence A is a syntactic consequence of Σ if there exists a proof from sentences in Σ to A. This is written as:*

$$\Sigma \vdash A \qquad (\Sigma \text{ 'proves' } A)$$

- *Sentence B is a semantic consequence of Σ if for every truth assignment to variables that makes all sentences in Σ true, sentence B must be true under that assignment as well. This is written as:*

$$\Sigma \vDash B \qquad (\Sigma \text{ 'entails' } B)$$

From this, we can give more concrete definition to a formal theory:

Definition 1.1.3. ***Formal Theory (Concrete Definition)***
A formal theory is a set of sentences Σ closed under semantic consequence. That is, for sentence X,

$$\textit{if } \Sigma \vDash X, \textit{ then } X \in \Sigma.$$

Interchangeably, we have:

$$\textit{if } \Sigma \vDash X, \textit{ then } \Sigma \vdash X.$$

Informally, this means if X is true under our interpretation of Σ's knowledge, then Σ can prove X using these knowledge.

1.2 Properties of Formal Theory

——————————Helper Definitions ——————————

Definition 1.2.1. ***Decidable Set***
A set of natural numbers S is called decidable (or computable) if given a number n, there exists an algorithm (a finite sequence of procedures) that terminates after finite time, and correctly decides whether the claim "$n \in S$" is true, or false.

Definition 1.2.2. ***Semi-Decidable Set***
Semi-decidable (or computably enumerable) sets are more general than decidable sets. A set S is semi-decidable if there exists an algorithm that enumerates the members of S.

As such, given a number n, we can run the algorithm over S, only stopping when we see n. If $n \in S$, then we know algorithm terminates in finite time, and decides the claim "$n \in S$" to be true. Otherwise, the algorithm may run forever (say S is a countable infinite set).

Next, we have the idea of *properties* and *relations* among numbers. For example,

- A property $P(x)$ for natural number x could be "x is even".
- A relation $R(x, y, z)$ for natural numbers x, y, z (denoted as $\vec{x}$) could be "$x + y = z$".

For the sake of simplicity, in general we treat property as a special kind of relation (a unary relation). Also, we can carry over the notion of decidable / semi-decidable sets to these relations, specifically, we have: ↓

Definition 1.2.3. ***Decidable Relation***
A relation R over domain D is decidable iff there exists an algorithm such that for every object $o \in D$, the algorithm can determine if o has relation R or not.

Remark. *To make a connection, observe that:*

R is a decidable relation $\iff$ $\{o \in D$: o has relation $R\}$ is a decidable set

Definition 1.2.4. ***Semi-Decidable Relation***
A relation R over domain D is semi-decidable iff there exists an algorithm such that for every object $o \in D$, the algorithm outputs:

- *True, if o has relation R*
- *Undefined (i.e. algorithm doesn't halt), if o does not have relation R*

Definition 1.2.5. ***Effectively Formalized Language***
A language L is effectively formalized iff:

1. *Its alphabet is a finite set.*
2. *Its syntactic properties are all decidable. That is, given a string, it is decidable if it is a term, a wff, or a sentence.*

3. *We are able to determine the unique truth-condition (True or False) of any sentence according to our semantics.*

Definition 1.2.6. ***Effectively Axiomatized Formal Theory*** *A formal theory is effectively axiomatized if it has an effectively formalized language $\mathcal{L}$, and its set of axioms, and its set of inference rules, are decidable sets.*

The key idea here is that in a effectively axiomatized formal theory, we can correctly determine if a formula is an axiom or not, ALSO, we can correctly determine if a proof – a sequence of *wff*s, follows our rules of inference or not.

Definition 1.2.7. ***Formal System*** *A formal system is an effectively axiomatized formal theory.*

Definition 1.2.8. ***Consistent (Formal Theory)*** *A formal theory is said to be consistent if it cannot derive contradictory statements. That is, we cannot derive both a statement and its negation in our system.*

Definition 1.2.9. ***Complete (Formal Theory)*** *A formal theory is complete if it can formally decide every wff of its language. In other words, for every grammatical statement in the theory's language, either the theory can prove it to be true, or it can prove its negation to be true.* [1]

Coming back to our analogy of 'expert system', we ask the question: does there exist a day when we feed so much knowledge into an expert system's knowledge base, such that our machine can answer, quite literally, any question we can possibly formulate? That is, can we ever get a complete expert system?

[1] This is different from the idea of "*Completeness*" with respect to semantics, which means everything true can be proven by the proof system.

1.3 Assuming Soundness

We will assume that our expert system is "*sound*", that is:

Definition 1.3.1. ***Soundness***
A formal theory Σ is sound iff everything it proves is true, this means for any sentence φ:

$$\Sigma \vdash \varphi \rightarrow \Sigma \vDash \varphi$$

This property is very close to the idea of *Consistency* with a slight twist, as for example, a consistent formal theory doesn't have to be sound (it can just consistently prove the wrong stuff). The key observation is that if we don't have this soundness property, then the proof system we have at hand is garbage, since it can prove things which we know aren't true!

Chapter 2

Language to do Basic Arithmetic

2.1 The $\mathcal{L}_{\mathcal{A}}$ Language

A formal system smart enough should be able to carry out basic arithmetic, that is, it can prove things about natural numbers. So our formal system should contain a language rich enough to make corresponding expressions.

For this purpose, we introduce the first-order language of basic arithmetic – $\mathcal{L}_{\mathcal{A}}$, it has the following non-logical symbols, along with their interpretations:

Symbol	**Meaning**
0	the constant zero
S	successor function
$+$	addition function
$\times$	multiplication function

One thing to note is the successor function; in the natural numbers we have 1 being the successor of 0, hence $S0 = 1$. Going onwards, we have $SS0 = 2$, $SSS0 = 3$, etc. For notational convenience, we use $\overline{n}$ to represent $SSS...S0$ with n occurrences of S.

Besides the non-logical symbols, $\mathcal{L}_{\mathcal{A}}$ also inherits all logical vocabulary from first-order logic, namely:

- **Quantifiers**: $\forall$ for "for all", $\exists$ for "exists"
- **Connectives**: $\rightarrow$ for "implies", $\wedge$ for "and", $\vee$ for "or", $\neg$ for "not"
- **Parentheses and punctuation**: ( ) ,
- **An infinite set of variables**: $x_1, x_2, ...$
- **Equality predicate**: $=$

By restricting the domains of the quantifiers in $\mathcal{L}$ to only natural numbers, we get a very elegant language capable of expressing a good amount of arithmetic over the natural numbers.

For notational convenience, we refer to formulas, *wff*s and sentences in the language of $\mathcal{L}_\mathcal{A}$ as "$\mathcal{L}_\mathcal{A}$-formulas", "$\mathcal{L}_\mathcal{A}$-*wff*s", and "$\mathcal{L}_\mathcal{A}$-sentences", respectively.

2.2 Robinson Arithmetic

Robinson Arithmetic (denoted as **Q**) is a theory that uses the language $\mathcal{L}_\mathcal{A}$. It has the following 7 assumptions for its axioms:

- $\forall x(Sx \neq 0)$ (No number has 0 as its successor)
- $\forall x \forall y(Sx = Sy \rightarrow x = y)$ (2 numbers having the same successor implies they are the same number)
- $\forall x(x \neq 0 \rightarrow \exists y(x = Sy))$ (a non-0 number is always an successor of some number)
- $\forall x(x + 0 = x)$ (addition with 0)
- $\forall x(x \times 0 = 0)$ (multiplication with 0)
- $\forall x \forall y(x + Sy = S(x + y))$ (simple addition)
- $\forall x \forall y(x \times Sy = (x \times y) + x)$ (simple multiplication)

Definition 2.2.1. ***Robinson Arithmetic (Q)***
Q *is the formal theory containing the formal language $\mathcal{L}_{\mathcal{A}}$, the above 7 axioms, plus first-order logic as its rule of inference.*

We will take it for granted that **Q** is a **sound** theory, since its axioms are all true under our way of interpreting the natural numbers. Further, using **Q** as a foundation, we can also build up more complex of theories!

Definition 2.2.2. *For formal theories $\mathcal{T}, \mathcal{T}'$, we say $\mathcal{T}'$ is an extension of $\mathcal{T}$, or $\mathcal{T}'$ contains $\mathcal{T}$, iff:*

$$\mathcal{T} \subseteq \mathcal{T}'$$

Remark. *As an extension to **Q**, we will obtain the Peano Arithmetic (denoted **PA**), if we also include the axiom of induction to **Q**'s axioms:*

$$\phi(0) \wedge \forall x[\phi(x) \rightarrow \phi(Sx)] \rightarrow \forall x \phi(x)$$

We are very interested in **Q** in the context of the First Incompleteness Theorem; as a matter of fact, what we meant with being able to do 'elementary arithmetic' back in Section 1, is actually just being able to do everything that **Q** can do, and this leads to a more detailed preview on Gödel's First Incompleteness Theorem:

*Every consistent formal system that contains **Q** must be incomplete.*

2.3 Strength of Q

Following directly from the previous section, the reason why **Q** is involved here, is because we can deduce an incredibly powerful theorem (with super long proof). First, some definitions:

——————Important Def on Quantifier Complexity ——————

Other than the standard "∀" and "∃" quantifiers, we may also want to restrict our variables further to a certain bound. To do so, we first introduce the '≤' symbol:

Definition 2.3.1. *Let x, y be terms, then:*

$$\boldsymbol{x \leq y} \text{ stands for } \exists z(x + z = y)$$

This leads us to the notion of **bounded quantifiers**, namely:

$$\forall x(x \leq b \to P(x)) \quad \text{and} \quad \exists x(x \leq b \to P(x))$$

Where b specifies the bound and $P(x)$ specifies the property; and we can further abbreviate these sentences as:

$$(\forall x \leq b)P(x) \quad \text{and} \quad (\exists x \leq b)P(x)$$

Definition 2.3.2. *A formula in language $\mathcal{L}_{\mathcal{A}}$ is bounded iff all its quantifiers are bounded. We call such formula* $\boldsymbol{\Delta_0}$ ***formula***.

With this notion in hand we can introduce something stronger:

Definition 2.3.3. *A* $\boldsymbol{\exists\Delta_0}$ ***formula*** *(also called Σ_1 formula) is one with the form $\exists y A$, where A is a Δ_0 formula.*

Following that, we define the notion of representability:

Definition 2.3.4. ***Strong Representation***
In formal theory $\mathcal{T}$, a formula $\varphi(\vec{x})$ strongly represents an n-ary relation over the natural numbers R iff for any $\vec{x} \in \mathbb{N}^n$:

- *If $\vec{x}$ has relation R, then $\mathcal{T} \vdash \varphi(\overline{\vec{x}})$*
- *If $\vec{x}$ does not have relation R, then $\mathcal{T} \vdash \neg\varphi(\overline{\vec{x}})$*

Definition 2.3.5. ***Weak Representation***
In formal theory $\mathcal{T}$, a formula $\varphi(\vec{x})$ weakly represents an n-ary numerical relation R iff for any $\vec{x} \in \mathbb{N}^n$:

$$\vec{x} \text{ has relation } R \leftrightarrow \mathcal{T} \vdash \varphi(\overline{\vec{x}})$$

This leads to the incredible theorem that will be very useful later in our attempt to prove the First Incompleteness Theorem:

Theorem 2.3.1. ***Representability Theorem***
*Every semi-decidable relation is weakly representable by some $\exists\Delta_0$ formula in **Q** (and hence all consistent extensions of **Q**)*

Remark. *The theorem also states that every decidable relation is strongly representable in **Q**. But in the context of proving the first incompleteness theorem, this is not required.*

2.4 Representability Theorem Proof

This section presents a proof for the Representability Theorem mentioned above. The proof is fairly technical, and acts only as a way to convince you that the theorem is true. To those uninterested in the technical details, it is totally fine to skim through this section or even skip it entirely.

To outline the proof, we first dedicate a few sub-sections to definitions and helper-theorems. In essence, the proof for the Representability Theorem relies on two pillars, the Main Lemma (Lemma 2.4.5), and the Exists Delta Theorem (Theorem 2.4.12).

2.4.1 What Makes Q Special?

The axioms of **Q** all exist for a reason, as a matter of fact they can prove some neat things! Suppose we have a formula A of the form "$((\overline{2} + \overline{1}) + \overline{3})$", then we claim **Q** can correctly evaluate A, and prove that $A = \overline{6}$.

Lemma 2.4.1. ***Q*** *can correctly evaluate arithmetic operations* $(\overline{n}+\overline{m})$ *and* $(\overline{n} \times \overline{m})$

Proof. First we check the case for $(\overline{n}+\overline{m})$. We proceed using induction on $\overline{m}$:

Base Case: Simply by axiom 4, we have $(\overline{n}+0) = \overline{n}$

Induction Step: We next suppose that $(\overline{n} + \overline{k-1})$ can be correctly evaluated, say the evaluated number is $\overline{v}$; we want to show that $(\overline{n}+\overline{k})$ can also be correctly evaluated:

$$\begin{aligned}(\overline{n}+\overline{k}) &= (\overline{n} + S(\overline{k-1})) \\ &= S(\overline{n} + \overline{k-1}) && \text{(By axiom 6)} \\ &= S\overline{v} && \text{(By induction hypothesis)} \\ &= \overline{v+1}\end{aligned}$$

So we are able to evaluate $(\overline{n} + \overline{k})$ for arbitrary k. Therefore we can certainly evaluate $(\overline{n}+\overline{m})$. On the other hand, the case for $(\overline{n}\times\overline{m})$ is similar using axioms 5 and 7. □

Lemma 2.4.2. *Let A be an arithmetic operation in $\mathcal{L}_{\mathcal{A}}$, and suppose A evaluates to number n, then:*

$$\boldsymbol{Q} \vdash (A = \overline{n})$$

That is, ***Q*** *proves that A evaluates to n*

Proof. As an arithmetic operation, A is built up from smaller sub-sentences of the form $(\overline{n}+\overline{m})$ or $(\overline{n}\times\overline{m})$ or $\overline{n}$, chained together using more '+' and '×' symbols. As an example we could have:

$$(\overline{n_1} \times \overline{m_1}) + ((\overline{n_2} + \overline{m_2}) \times \overline{n_3})$$

We use structural induction in this case:

Base Case: The simplest arithmetic operations $(\overline{n} + \overline{m})$ and $(\overline{n} \times \overline{m})$ can both be evaluated by **Q** by Lemma 2.2.1., and $\overline{n}$ is evaluated trivially.

Induction Step: Suppose A_1 and A_2 are both arithmetic operations that can be evaluated by **Q** to $\overline{v_1}$ and $\overline{v_2}$, respectively; we want to show that $(A_1 \times A_2)$ and $(A_1 + A_2)$ can also be evaluated by **Q**.

Simply, we have $(A_1 \times A_2) = (\overline{v_1} \times \overline{v_2})$ and $(A_1 + A_2) = (\overline{v_1} + \overline{v_2})$, and these can be evaluated by **Q** by Lemma 2.2.1., so we may conclude that any arithmetic operation that is built up from simpler ones, can too be evaluated by **Q**.

□

Theorem 2.4.3. ***Q*** *can decide every quantifier-free $\mathcal{L}_\mathcal{A}$ sentence. That is, for every quantifier-free sentence φ in $\mathcal{L}_\mathcal{A}$:*

- *If φ is true, then **Q** proves φ*
- *If φ is false, then **Q** proves $\neg\varphi$*

Proof. Let φ be an arbitrary sentence in $\mathcal{L}_\mathcal{A}$ without any quantifiers, we proceed using structural induction:

The only predicate in $\mathcal{L}_\mathcal{A}$ (such that a sentence can be evaluated to be true or false), is the equality predicate '='. So sentence φ is a sequence of sub-sentences of the form '$A = B$', linked together using logical connectives, where A, B are arithmetic operations.

Base Case: Want to show that for sentence "$A = B$", where A, B are arithmetic operations, if the sentence is true, then **Q** proves it to be true. Otherwise **Q** proves it to be false.

By Lemma 2.2.2., we know **Q** can individually, and correctly evaluate A and B. We also know that the equality predicate '=' will output True iff the evaluated values on both sides are the same. So:

- If "$A = B$" is true, then the evaluated values of A, B are the same, hence the equality predicate will output True, so **Q** proves "$A = B$" to be true.

- Otherwise, the evaluated values of A and B are not the same, so the equality predicate will output False, so $\mathbf{Q}$ proves its negation to be true, as desired.

Induction Step: The logical connectives are a part of inference rules of $\mathbf{Q}$, so any sentence built up from simpler ones using connectives can also be proven by $\mathbf{Q}$, so long as the smaller sub-sentences (or atomic sentences) can be proven to be true or false. □

2.4.2 The Main Lemma

Up next, we want to know if $\mathbf{Q}$ can decide the $\exists\Delta_0$ formulas. As a reminder, these formulas contain quantifiers of the form $(\exists x \leq b)$ and $(\forall x \leq b)$. To answer this, it is first natural to ask whether $\mathbf{Q}$ can prove facts involving the relation "$x \leq y$":

Theorem 2.4.4. *The relation "$x \leq y$" is strongly represented in $\mathbf{Q}$ by the formula $\exists z(\overline{x} + z = \overline{y})$*

Proof. Recall the notion of representation back in section 2.3, we want to show that:

"$x \leq y \rightarrow \mathbf{Q} \vdash \exists z(\overline{x} + z = \overline{y})$" and "$x > y \rightarrow \mathbf{Q} \vdash \neg\exists z(\overline{x} + z = \overline{y})$"

- We first start with the first one, suppose $x \leq y$, then indeed there exists k such that $x + k = y$, and by Lemma 2.2.2., we know:

$$\mathbf{Q} \vdash (\overline{x} + \overline{k} = \overline{y})$$

It immediately follows, by first-order logic, that:

$$\mathbf{Q} \vdash \exists z(\overline{x} + z = \overline{y})$$

- We next consider the second one. Suppose $x > y$, and for the sake of contradiction, suppose $\mathbf{Q}$ proves $\exists z(\overline{x} + z = \overline{y})$, by first-order logic, it naturally follows that for some number k, $\mathbf{Q}$ proves $\overline{x} + \overline{k} = \overline{y}$. But by $x > y$ and the fact $k \geq 0$ (as k is the k-th successor of 0):

$$x + k \geq x > y$$

Therefore, $x + k \neq y$, so by Theorem 2.2.3., **Q** proves $\neg(\overline{x} + \overline{k} = \overline{y})$, leading to contradiction!

□

This means our definition suffices! The relation "$x \leq y$" is equivalent to "$\exists z(x + z = y)$" in **Q**. In particular, this means **Q** can prove bounded formulas, or Δ_0 formulas! We just have to translate every "$x \leq y$" we see into its representative "$\exists z(x + z = y)$".

Next, observe that for any number n:

- If $\mathbf{Q} \vdash (\varphi(\overline{0}) \wedge \varphi(\overline{1}) \wedge ... \wedge \varphi(\overline{n}))$, then $\mathbf{Q} \vdash (\forall x \leq \overline{n})\varphi(x)$
- If $\mathbf{Q} \vdash (\varphi(\overline{0}) \vee \varphi(\overline{1}) \vee ... \vee \varphi(\overline{n}))$, then $\mathbf{Q} \vdash (\exists x \leq \overline{n})\varphi(x)$

With these facts altogether, we claim:

Lemma 2.4.5. ***(Main Lemma) Q*** *can decide any Δ_0 sentence. That is, it can prove the true Δ_0 sentences, and disprove the false ones (by proving their negations to be true).*

Proof. Let S be a bounded sentence, move all its negations, or '$\neg$' symbols inwards, and modify the connectives and quantifiers accordingly, specifically:

- $\neg(A \vee B)$ becomes $\neg A \wedge \neg B$
- $\neg(A \wedge B)$ becomes $\neg A \vee \neg B$
- $\neg(\forall x \leq b)A$ becomes $(\exists x \leq b)\neg A$
- $\neg(\exists x \leq b)A$ becomes $(\forall x \leq b)\neg A$

We then proceed by structural induction [1]:

[1] Note we do not include the connective "$\rightarrow$" because it can be represented using other connectives. That is, "$A \rightarrow B$" is equivalent to "$\neg A \vee B$".

Base Case: The simplest bounded sentences actually have no quantifiers. The fact that all their quantifiers are bounded is *vacuously* true. We also note that all negations are driven inwards. Therefore, our base cases are simply:

$$\overline{x} = \overline{y} \text{ and } \overline{x} \neq \overline{y}, \text{ for arbitrary } x, y$$

These, by Theorem 2.4.3, can be decided by **Q**.

Induction Step: Suppose S has all its negation symbols driven inwards, and does not satisfy the base case. Also assume that the simpler bounded sentences are all provable by **Q**. We know S has a main connective or an outermost quantifier, that is, S takes up the form:

$$\text{“}A \wedge B\text{”}, \text{ “}A \vee B\text{”}, \text{ “}(\exists x \leq \overline{n})A\text{”}, \text{ or “}(\forall x \leq \overline{n})A\text{”}$$

- Connectives: For sentences

  $$\text{“}A \vee B\text{” and “}A \wedge B\text{”},$$

  We can decide them simply by the truth values of A and B.

- Bounded Quantifiers: For sentences

  $$\text{“}(\exists x \leq \overline{n})A(x)\text{” and “}(\forall x \leq \overline{n})A(x)\text{”},$$

  we can use the just-introduced technique to break apart the bounded quantifiers, and convert them into more convenient expressions:

  $\text{“}A(0) \vee A(\overline{1}) \vee ... \vee A(\overline{n})\text{”}$ and $\text{“}A(0) \wedge A(\overline{1}) \wedge ... \wedge A(\overline{n})\text{”}$, respectively.

  They are decidable by **Q** since by assumption, the simpler sentences $A(\overline{i})$ are decidable, for each $i \leq n$.

□

This leads to our desired corollary:

Corollary 2.4.5.1. ***Q*** *can prove any true* $\exists\Delta_0$ *sentence.*

Proof. Let "$\exists x A(x)$" be a true $\exists\Delta_0$ sentence. Since it is true, there exists some number k such that $A(k)$ holds.

We know $A(k)$ is a Δ_0 formula, so $\mathbf{Q}$ can prove it by the Main Lemma. Therefore, $\mathbf{Q} \vdash A(\overline{k})$, and it naturally follows that $\mathbf{Q} \vdash \exists x A(x)$ using first-order logic. □

2.4.3 Primitive Recursive Functions

In this sub-section, we introduce a ery special kind of functions called the primitive recursive (p.r.) functions. These are in essense, the functions that can be computed by a computer program using only bounded loops, that is, "for" loops.

Definition 2.4.1. ***P.R. Functions***
A function $f(x_1, ..., x_n)$ is called primitive recursive if either:

- *f is the zero function*
- *f is the successor function S*
- *f is the projection function [2], that is, for some index i:*

$$f(x_1, ..., x_n) = x_i$$

- *f is **defined by composition** of primitive recursive functions. That is, if $g(x_1, ..., x_m)$ and $h_1(x_1, ..., x_n), ..., h_m(x_1, ..., x_n)$ are all primitive recursive, then*

$$f(x_1, ..., x_n) = g(h_1(x_1, ..., x_n), ..., h_m(x_1, ..., x_n))$$

  is also primitive recursive.

- *f is **defined by recursion** of two primitive recursive functions. That is, if $g(x_1, ..., x_{n-1})$ and $h(x_1, ..., x_{n+1})$ are both primitive recursive, then the function f recursively defined by:*

  1. $f(x_1, ..., x_{n-1}, 0) = g(x_1, ..., x_{n-1})$ *(Base Case)*

[2] This can be seen as similar to list-indexing in your favorite programming language

2. $f(x_1, ..., x_{n-1}, k+1) = h(x_1, ..., x_{n-1}, k, f(x_1, ..., x_{n-1}, k))$ *(Recursion)*

is also primitive recursive.

Now, there is an important helper-theorem involving p.r. functions. First, there are some lemmas.

Lemma 2.4.6. ***(Closure Lemma)*** *The $\exists\Delta_0$ formulas are closed under $\exists, \vee, \wedge, (\forall \leq)$ and $(\exists \leq)$. This means for $\exists\Delta_0$ formulas A, B,*

$$\text{“}\exists xA\text{”},\ \text{“}A \vee B\text{”},\ \text{“}A \wedge B\text{”},\ \text{“}(\forall x \leq n)A\text{”} \text{ and } \text{“}(\exists x \leq n)A\text{”},$$

are also logically equivalent to $\exists\Delta_0$ formulas.

Proof. We will prove the case “$A \wedge B$”, and the other cases are similar.

For $A \wedge B$, say A takes the form $\exists zC(z)$ and B takes the form $\exists yD(y)$, then $A \wedge B$ is equivalent to $\exists z\exists yC(z) \wedge D(y)$. We can introduce an upper bound on z, y without any harm, since the upperbound is arbitrary:

$$\exists u(\exists z \leq u)(\exists y \leq u)C(z) \wedge D(y)$$

This formula is $\exists\Delta_0$, and we are done.

□

Definition 2.4.2. ***Graph of a Function***
For the k-ary function $f : \mathbb{N}^k \to \mathbb{N}$, define $graph(f)$ to be the $k+1$-ary relation:

$$R(\vec{x}, y) := (f(\vec{x}) = y)$$

Then we have the following result:

Lemma 2.4.7. ***(Exists Delta Lemma)*** *If f is a p.r. function, then for the relation $R := graph(f)$, there is some $\exists\Delta_0$ formula φ such that:*

$$R(\vec{x}, y) \text{ is true } \iff \varphi(\vec{x}, y) \text{ is true}$$

Proof. We begin the proof for $graph(f)$ using structural induction on the primitive recursive functions:

Base Case: We check out the initial functions:

- Zero Function: Relation $(Z(\vec{x}) = y)$, where Z is the zero function can be written as the formula $y = 0$.
- Successor Function: Relation $(S(x) = y)$ can be written as the formula $Sx = y$.
- Projection Function $I_i(\vec{x})$ (which returns the entry of $\vec{x}$ on index i): Relation $(I_i(\vec{x}) = y)$ can be written as the formula $y = x_i$.

Induction Step: Suppose $graph(g)$ and $graph(h)$ are all representable by $\exists\Delta_0$ formulas, want to show that the function f defined by the composition, or recursion of g, h, also have $graph(f)$ representable by some $\exists\Delta_0$ formula.

- Composition: $f(\vec{x}) = g(h_1(\vec{x}), ..., h_n(\vec{x}))$, so the relation $(f(\vec{x}) = y)$ is equivalent to:

$$\exists y_1 ... \exists y_n (h_1(\vec{x}) = y_1) \wedge ... \wedge (h_n(\vec{x}) = y_n) \wedge (g(y_1, ..., y_n) = y)$$

  By assumption, this becomes:

$$\exists y_1 ... \exists y_n D_1 \wedge ... \wedge D_n \wedge D_{n+1},$$

  where each D_i is an $\exists\Delta_0$ formula. Then by the Closure Lemma, this thing is also an $\exists\Delta_0$ formula.

- Primitive Recursion: Super hard! See next section for the whole proof. Take a break for now. :)

□

2.4.4 Proof Continued

To handle the primitive recursion case, we will have to go through some troubles first. In particular, we need the Chinese Remainder Theorem, its proof, however, is not included for the sake of compactness of this text.

Definition 2.4.3. ***(Gödel β Function)***

$$\beta(c, d, i) = c \mod (d(i+1)+1)$$

Theorem 2.4.8. ***Chinese Remainder Theorem (CRT)***
Given $r_0, ..., r_n$ and $m_1, ..., m_n$, such that for each i,

$$0 \leq r_i < m_i,$$

and each m_i, m_j is pairwise coprime (their greatest common divisor = 1), then there exists one and only one x, where $x < m_1 \times ... \times m_n$, such that for each i:

$$x \mod m_i = r_i$$

Lemma 2.4.9. *For any sequence of numbers $r_1, ..., r_n$, there exists c, d such that for each $i = 0, ..., n$:*

$$\beta(c, d, i) = r_i$$

Proof. Let $d = (n + r_1 + ... + r_n + 1)!$ and each $m_i = d \times (i+1) + 1$.

Assume WLOG that $0 \leq i < j \leq n$, we claim that m_i, m_j are pairwise coprime. Since, assuming for the sake of contradiction, that there is some prime p that satisfies:

$$p \mid m_i \text{ and } p \mid m_j,$$

then this means:

$$p \mid (d \times (i+1)+1) \text{ and } p \mid (d \times (j+1)+1),$$

then jumping ahead to Lemma 3.2.1, this means p divides their difference $d \times (j - i)$.

But p cannot divide both d and $d(i+1)+1$, since again by Lemma 3.2.1, this means p divides 1.

So the only option left is p divides $(j-i)$, meaning $p \leq (j-i) < n$, so p divides d by the construction of d, leading to contradiction.

Therefore by CRT, there exists a unique number x such that:

$$\beta(x, d, i) = (x \mod m_i) = r_i$$

□

Lemma 2.4.10. *graph(β) is a 4-place relation R such that there ewxists some Δ_0 formula φ and:*

$$R(c, d, i, y) \text{ is true} \iff \varphi(c, d, i, y) \text{ is true}$$

Proof. For the relation $(\beta(c, d, i) = y)$, we can write it as the Δ_0 formula:

$$(\exists x \leq c)[(c = x[d(i+1)+1]) \wedge y \leq d(i+1)]$$

□

Now, we are ready to handle the last step of the induction:

Proof. (Proof Continued) Recall we say f is defined by primitive recursion from g and h iff:

- $f(\vec{x}, 0) = g(\vec{x})$
- $f(\vec{x}, k+1) = h(\vec{x}, k, f(\vec{x}, k))$

So $f(\vec{x}, k) = y$ iff there are $r_0, ..., r_n$, such that:

- $r_0 = g(\vec{x})$ (base case)
- $r_{i+1} = h(\vec{x}, i, f(\vec{x}, i))$ (recursive case)
- $r_k = y$ (equality to y)

Therefore by Lemma 2.4.9, we can represent the relation $f(\vec{x}, k) = y$ as a formula as follows:

$$\begin{gathered}\exists c \exists d[(\beta(c,d,0) = g(\vec{x})) \\ \wedge \\ [(\forall i \leq (k-1))\beta(c,d,i+1) = h(\vec{x},i,\beta(c,d,i))] \\ \wedge \\ \beta(c,d,k) = y]\end{gathered}$$

Next, observe that:

- $(\beta(c,d,0) = g(\vec{x}))$ can be written as:

$$\exists w(\beta(c,d,0) = w) \wedge (g(\vec{x}) = w)$$

- $\beta(c,d,i+1) = h(\vec{x},i,\beta(c,d,i))$ can be written as:

$$\exists w \exists z(\beta(c,d,i+1) = w) \wedge (w = h(\vec{x},i,z) \wedge (z = \beta(c,d,i))$$

We know $graph(g)$ and $graph(h)$ can all be written as $\exists\Delta_0$ formulas by the induction hypothesis, and by Lemma 2.4.10, we know $graph(\beta)$ can be written as a Δ_0 formula. Therefore, we can conclude by the Closure Lemma 2.4.6 that the above two bullets are also $\exists\Delta_0$ formulas.

Then again by the Closure Lemma 2.4.6, the entire formula expressing the relation $f(\vec{x},k) = y$ is an $\exists\Delta_0$ relation. Again, if this is true, then **Q** proves it by Corollary 2.4.5.1, and if **Q** proves the formula to be true, then by soundness, the formula is true. So this $\exists\Delta_0$ formula weakly represents the relation $f(\vec{x},k) = y$.

□

2.4.5 The Exists Delta Theorem

Great! We just finished the long, hard proof for the Exists Delta Lemma, and now we can focus on the powerful result from this lemma. We begin by giving a rather abstract definition to the semi-decidable relations:

Definition 2.4.4. ***Semi-Decidable Relation (Abstract Ver.)*** *A relation R is semi-decidable iff there is a computable function $g(\vec{x})$ such that $R(\vec{x})$ holds $\iff \vec{x} \in Domain(g)$.*

This essentially means whenever $R(\vec{x})$ is true, the function g can compute input $\vec{x}$ to inform us that $\vec{x}$ works for R, and whenever $R(\vec{x})$ is false, $g(\vec{x})$ could be undefined.

Definition 2.4.5. ***P.R. Predicate***
A p.r. predicate is a predicate P whose truth value can be determined by a p.r. function. The function outputs 1 *if P is true, and* 0 *otherwise.*

Theorem 2.4.11. ***Normal Form Theorem***
A relation R is semi-decidable iff there is a p.r. predicate $P(\vec{x}, y)$, such that $R(\vec{x})$ holds $\iff \exists y P(\vec{x}, y)$

Proof. (Proof Idea)
"$\Leftarrow$": Suppose there is a p.r. predicate $P(\vec{x}, y)$ s.t. $R(\vec{x}) \iff \exists y P(\vec{x}, y)$, then define $g(\vec{x}) := \mu y R(\vec{x}, y)$, where '$\mu$' is the minimization operator which searches for the smallest y satisfying the relation $R(\vec{x}, y)$, and return it as output. Then:

$$R(\vec{x}) \iff \vec{x} \in Domain(g)$$

"$\Rightarrow$": Suppose R semi-decidable, then there is computable function g such that $R(\vec{x}) \iff \vec{x} \in Domain(g)$. Then let G be a program to compute g, and define:

$$P(\vec{x}, y) := G(\vec{x}) \text{ is defined in } y \text{ steps}$$

This relation is a p.r. predicate, since we can just have a program that lists out the first y steps that $G(\vec{x})$ takes, and conclude whether $G(\vec{x})$ returns or not. This is purely mechanical. (Recall that p.r. functions are the ones that can be computed by a program whose loops are all bounded loops, and in this case the looping upperbound is y).

As such, we get a p.r. predicate P so that:

$$R(\vec{x}) \leftrightarrow \exists y P(\vec{x}, y)$$

□

And the theorem follows:

Theorem 2.4.12. ***Exists Delta Theorem***
Every semi-decidable relation R has an equivalent $\exists\Delta_0$ formula $\varphi(\vec{x})$ such that:

$$R(\vec{x}) \text{ is true} \iff \varphi(\vec{x}) \text{ is true}$$

Proof. For semi-decidable relation R, by the Normal Form Theorem 2.4.11, $R(\vec{x})$ is true $\iff \exists y P(\vec{x}, y)$, where P's truth can be computed by a primitive recursive function that outputs 1 (for True) if P is true, and 0 (for False) if P is false.

Let f be the primitive recursive function that does this computation, then:

$$\begin{aligned} P(\vec{x}, y) \text{ is true} &\iff f(\vec{x}, y) = 1 \\ &\iff R(\vec{x}, y, 1) \\ &\quad \text{(where } R \text{ is the relation } graph(f) \text{ as in Def 2.4.2)} \\ &\iff \varphi(\vec{x}, y, 1) \\ &\quad \text{(where } \varphi \text{ is an } \exists\Delta_0 \text{ formula by the Exists Delta Lemma 2.4.7)} \\ &\iff \varphi_2(\vec{x}, y) \\ &\quad \text{(Set the '1' in } \varphi(\vec{x}, y, 1) \text{ to be a constant to get } \varphi_2(\vec{x}, y)) \end{aligned}$$

So in sum, we have:

$$R(\vec{x}, y) \text{ is true} \iff \exists y P(\vec{x}, y) \iff \exists y \varphi_2(\vec{x}, y)$$

Where $\varphi_2(\vec{x}, y)$ is an $\exists\Delta_0$ formula. Then by the Closure Lemma 2.4.6, $\exists y \varphi_2(\vec{x}, y)$ is also an $\exists\Delta_0$ formula, we are done! □

2.4.6 Representability Theorem Conclusion

What's left is very short, we want to show that every semi-decidable relation is weakly representable in **Q** by an $\exists\Delta_0$ formula.

Proof. Suppose there is a recursive relation $R(\vec{x})$, then by the Exists Delta Theorem 2.4.12,

$$R(\vec{x}) \text{ is true} \iff \text{some } \exists\Delta_0 \text{ formula } \varphi(\vec{x}) \text{ is true}$$

Then by Corollary 2.4.5.1, if $\varphi(\vec{x})$ is true then $\mathbf{Q}$ can prove it. On the other hand, if $\mathbf{Q}$ proves $\varphi(\vec{x})$, then by soundness of $\mathbf{Q}$, $\varphi(\vec{x})$ must be true. As such we get:

$$R(\vec{x}) \text{ is true} \iff \mathbf{Q} \vdash \varphi(\vec{x}),$$

which is just the definition of weak representation in $\mathbf{Q}$. □

Remark. *Besides the theory* $\boldsymbol{Q}$*, observe that this theorem also applies to all formal theories that are sound extensions of* $\boldsymbol{Q}$*.*

Chapter 3

Gödel Numbering

Our ultimate goal is to mechanically code sentences, and sequences of sentences into numbers such that they can be recognized and processed by our proof system; similar to the ASCII code in our modern computers.

3.1 Numbering Symbols

Consider our standard language of arithmetic $\mathcal{L}_{\mathcal{A}}$. For each symbol we see, we can assign it with a unique Gödel number. The following table is one possible assignment:

Symbol	**Gödel Number**	**Meaning**
0	1	zero
S	2	successor function
$\neg$	3	not
$\vee$	4	or
$\wedge$	5	and
$\exists$	6	exists
$\forall$	7	for all
$\rightarrow$	8	implies
$($	9	left parenthesis
$)$	10	right parenthesis
$=$	11	equals
$+$	12	addition
$\times$	13	times
x_i	$14 + i$	variables

One thing to note is that this mapping is not unique. As a matter of fact, there are infinitely many mappings possible.

3.2 Numbering Strings

Suppose we now have a sentence in our system; it is a sequence of symbols of length k. We want to code this expression into its own unique number, based on how we coded primitive symbols. To do so, we consider the prime numbers:

3.2.1 Some Number Theory

Recall that *prime numbers* are numbers greater than 1, and their only divisors are 1 and themselves. The first few prime numbers are 2, 3, 5, 7, 11, 13, 17... We claim there are infinitely many such prime numbers.

Lemma 3.2.1. *If an integer divides two integers a, b, then that integer also divides their difference $a - b$.*

Proof. Let $c \in \mathbb{Z}$ such that $c \mid a$ and $c \mid b$, then we have $m, n \in \mathbb{Z}$ so that $a = mc$ and $b = nc$. So:

$$\begin{aligned} a - b &= mc - nc \\ &= c(m - n) \\ &= cx && (\text{Let } x = m - n \in \mathbb{Z}) \end{aligned}$$

We know that $c \mid cx$ for $x \in \mathbb{Z}$, so $c \mid a - b$.

□

Theorem 3.2.2. *There are infinitely many prime numbers.*

Proof. Suppose there are only finitely many primes: $p_1, ..., p_n$, we take the product $P = p_1 p_2 ... p_n$, and let $Q = P + 1$.

- If Q is prime, then we just found a prime number that's greater than each p_i, so Q is not in our list of primes, contradiction!
- If Q is not prime, then it must have a prime divisor p_i. However, note that p_i also divides P (product of all primes). So by **Lemma 1**, p_i must also divide $Q - P = 1$. yet, we know $p_i > 1$, contradiction!

□

Theorem 3.2.3. ***(Fundamental Theorem of Arithmetic)*** *For every natural number $n > 1$, it can be written as a UNIQUE product of prime numbers.*

Proof. **Existence:**
We want to show every integer ≥ 2 is either a prime, or a product of primes. We do so using strong induction:

- Base Case: 2 is prime
- Induction Step: for every $n > 2$, if n is prime, then we are done. Suppose n isn't prime, then there exist a, b such that $n = ab$ and $1 < a, b < n$ (by property of composite numbers). Knowing that a, b are either primes, or product of primes by induction hypothesis, n must also be a product of primes.

Uniqueness:
Suppose there exist numbers such that their prime factorization are not unique. Then let s be the smallest such number. We have:

$$s = p_1 p_2 ... p_k = q_1 q_2 ... q_k,$$

where each p_i, q_j are primes, and $p_i \neq q_j$ for every i, j, otherwise s/p_i becomes a smaller number with non-unique prime factorization.

Assume WLOG that $p_1 < q_1$, and let $P = p_2 ... p_k$, $Q = q_2 ... q_k$. Then

$$\begin{aligned} s - p_1 Q &= q_1 Q - p_1 Q \\ &= (q_1 - p_1) Q \\ &< s \end{aligned}$$

Since s and p_1Q both contain the factor p_1, there's $p_1 \mid (q_1 - p_1)Q$. Since $(q_1 - p_1)Q < s$ and we assumed s to be the smallest number with non-unique factorization, the factor p_1 must exist, and it occurs either in the factorization of $(q_1 - p_1)$, or that of Q.

- Case 1: p_1 is a factor of $(q_1 - p_1)$:
  This is impossible as $0 < q_1 - p_1 < p_1$, so $p_1 \nmid (q_1 - p_1)$
- Case 2: p_1 is a factor of Q:
  This is also impossible since $q_i \neq p_1$ for all i, as shown above, yet $Q = q_2...q_k$.

As all cases lead to contradiction, we are done! □

3.2.2 Numbering a Formula

Let p_i be the i-th smallest prime number, so there is:

$$p_1 = 2, p_2 = 3, p_3 = 5, ...$$

Then given a formula of arbitrary length, say it has k symbols:

$$x_1 x_2 ... x_k$$

we first mechanically translate each symbol in the formula into its Gödel number based on our previous mapping. As such, we get a k-length sequence of Gödel numbers:

$$g_1 g_2 ... g_k$$

By Theorem 2, we know there is always enough primes to construct a prime sequence of length k:

$$p_1, p_2, ..., p_k$$

Then we can "code" the sequence of Gödel numbers into a number c, such that:

$$c = 2^{g_1} \times 3^{g_2} \times ... \times p_k^{g_k}$$

By the Fundamental Theorem of Arithmetic, we know c is uniquely factorized into primes, and by checking the powers on primes, we know c uniquely encodes a sequence of Gödel numbers. As such, we may encode any formula into some number c, and given some number c, we can decode it into its

corresponding formula.

For example, consider the formula "$\neg(0 = S0)$", meaning 0 doesn't equal to 1. We can encode it as follow:

$$\begin{array}{cccccccccc} & & \neg & & (& & 0 & & = & \\ c & = & 2^3\times & & 3^9\times & & 5^1\times & & 7^{11}\times & \end{array} \begin{array}{ccc} & &) \\ S & & \\ 11^2\times & & \\ 0 & & \\ 13^1\times & & \\ & & 17^{10} \end{array}$$

So we get a HUGE number:

$$c = 4,936,824,178,085,181,660,467,986,316,520$$

Remark. *The number itself doesn't matter. What matters is that we know we have a bijective map between numbers and formulas, so we can decode c back to "$\neg(0 = S0)$" simply using prime factorization. In essence, this technique allows us to interchange between numbers and formulas.*

Definition 3.2.1. ***Gödel Numbering***
Given a formula X as a sequence of symbols $x_1x_2...x_k$, and each x_i corresponds to Gödel number g_i. Let p_i be the i-th prime number, then the Gödel number for X is calculated as:

$$2^{g_1} \times 3^{g_2} \times ... \times {p_k}^{g_k}$$

3.2.3 Numbering Proofs

Next, we shift our attention to numbering proof, that is, a linear sequence of *wff*s. We can assign each proof with a *super Gödel number* in a very similar fashion:

Definition 3.2.2. ***Super Gödel Number***
Given a sequence of wffs $X_1, X_2, ..., X_n$ with their corresponding Gödel numbers $c_1, c_2, ..., c_n$. Let p_i be the i-th prime number, then the super Gödel number for the wff sequence $X_1, X_2, ..., X_n$ is:

$$2^{c_1} \times 3^{c_2} \times ... \times {p_n}^{c_n}$$

Chapter 4

First Incompleteness Theorem Proof

We are almost there! The hard work has all been laid out and we can finally tackle the main theorem (whose proof is surprisingly short and rather nice)! In this writing two proofs will be outlined; the first one is neat and a bit unconventional, and the second one is a classic.

4.1 Proof No.1

Definition 4.1.1. ***Decidable Theory***
A formal theory $\mathcal{T}$ is said to be decidable iff being a theorem of $\mathcal{T}$ is a decidable property.

Theorem 4.1.1. *Any consistent, complete formal system $\mathcal{T}$ is decidable.*

Proof. Given a sentence φ, we want to decide whether φ is a theorem in $\mathcal{T}$ or not.

Since $\mathcal{T}$ is effectively axiomatized, its set of theorems is a semi-decidable (or recursively enumerable) set, this means we can enumerate our theorems of the set, and start comparing them with then compare them with φ. (We can make the comparisons easily using Gödel numbers).

Then, since $\mathcal{T}$ is consistent and complete, eventually we must run into φ or $\neg\varphi$ (but not both) as we go along the list of theorems. Then if we saw φ, we conclude φ is indeed a theorem. Otherwise we can confidently say it is not a theorem. □

Theorem 4.1.2. *Any consistent formal system $\mathcal{T}$ containing **Q** cannot be decidable.*

Proof. Suppose, for the sake of contradiction, that there is a consistent formal system $\mathcal{T}$ containing **Q** that is decidable.

Then, notice that since $\mathcal{T}$ is effectively formalized, its set of *wff*s is a decidable set. Then, since all decidable sets are semi-decidable too, we can enumerate the set of *wff*s. Note that a simple algorithm may check if a *wff* is unary, so we can just enumerate the unary *wff*s:

$$\varphi_1(x), \varphi_2(x), \varphi_3(x), ...$$

We also introduce a nice unary relation $R(x)$, such that for any natural number n:

$$R(n) \text{ is true} \iff \mathcal{T} \vdash \neg\varphi_n(\overline{n})$$

Because $\mathcal{T}$ is decidable, for each n it can prove either $\varphi_n(\overline{n})$ or $\neg\varphi_n(\overline{n})$, so the relation $R(n)$ is decidable!

Then $R(n)$ is also semi-decidable, so by the Representability Theorem 2.3.1, this relation is weakly representable by some formula $\Psi(n)$ in $\mathcal{T}$. This means:

$$R(n) \text{ is true} \iff \mathcal{T} \vdash \Psi(\overline{n}) \tag{1}$$

This is great since as we keep on enumerating the unary *wff*s, we will eventually encounter the unary *wff* $\Psi(x)$. Say we find $\Psi(x)$ as the k-th *wff* we pop out. So $\varphi_k(x) \iff \Psi(x)$, but as such:

$$\begin{aligned} R(k) \text{ is true} &\iff \mathcal{T} \vdash \neg\varphi_k(\overline{k}) \\ &\iff \mathcal{T} \vdash \neg\Psi(\overline{k}) \end{aligned} \tag{2}$$

This brings contradiction between the lines labelled (1) and (2), which concludes our proof! □

Combining the two theorems, we get our desired result:

Theorem 4.1.3. ***Gödel's First Incompleteness Theorem***
Every consistent formal system containing $\boldsymbol{Q}$ *cannot be complete.*

Proof. Given a consistent formal system $\mathcal{T}$ containing $\mathbf{Q}$, by Theorem 4.1.2, it must not be decidable.

However, if we suppose $\mathcal{T}$ to be complete, then we know $\mathcal{T}$ is both consistent and complete, so by Theorem 4.1.1, $\mathcal{T}$ must be decidable, contradiction! So $\mathcal{T}$ cannot be a complete theory! □

4.2 Proof No.2

The second proof uses Gödel numbering and the idea of the diagonal lemma to construct a sentence that says: "I am not provable", and we use this sentence to arrive at the final result. Regarding the notation in this section, for arbitrary sentence D, we will use $\ulcorner D \urcorner$ to denote the corresponding Gödel number of D.

4.2.1 The Diagonal Lemma

Before the lemma itself, as a proof helper we first introduce the binary substitution function $sub(a, b)$, where a is the Gödel number of a formula $A(x)$, and b is some number. The function substitutes the number b into every occurrence of x in $A(x)$, and returns the resulting formula $A(b)$'s Gödel number. In short, we have:

$$sub(\ulcorner A(x) \urcorner, b) = \ulcorner A(\bar{b}) \urcorner$$

We can then construct the relation $R(x_1, x_2, x_3)$, which is true iff $sub(x_1, x_2) = x_3$ holds, that means there is formula $A(x)$ and number b, such that:

$$x_1 = \ulcorner A(x) \urcorner, x_2 = b, x_3 = \ulcorner A(\bar{b}) \urcorner$$

Notice that this relation $R(x, y, z)$ is decidable, since we can just mechanically decode x_1 into its corresponding formula , then substitute every

occurrence of x in this formula with the number x_2, then encode this formula and compare the resulting Gödel number with x_3. If equal, then the relation holds, otherwise it doesn't.

Now we are ready for the lemma:

Lemma 4.2.1. ***Diagonal Lemma***
For a formal theory $\mathcal{T}$ that contains $\boldsymbol{Q}$, let $A(x)$ be an arbitrary $\exists\Delta_0$ formula, then we can construct sentence D such that:

$$\mathcal{T} \vdash D \iff A(\ulcorner D \urcorner)$$

Proof. The key observation for this proof is that for any formula $B(x)$, we can always substitute in the Gödel number of $B(x)$ itself into $B(x)$, that being:

$$sub(\ulcorner B(x) \urcorner, \ulcorner B(x) \urcorner)$$

To start, we use the just-introduced relation $R(x_1, x_2, x_3)$, because it is decidable (and hence semi-decidable), by the Exists Delta Theorem 2.4.12, we have some $\exists\Delta_0$ formula φ such that:

$$R(x_1, x_2, x_3) \text{ is true} \iff \varphi(x_1, x_2, x_3) \text{ is true}$$

Next, given $\exists\Delta_0$ formula $A(x)$, we construct a new formula $B(x)$ with one free variable:

$$B(x) := \exists y(A(y) \wedge \varphi(x, x, y))$$

Suppose the Gödel number for $B(x)$ is k, that means $\ulcorner B(x) \urcorner = k$, then we can construct a new sentence D by substituting k into $B(x)$:

$$D := B(\overline{k}) := \exists y(A(y) \wedge \varphi(\overline{k}, \overline{k}, y))$$

Observe that since A and φ are both $\exists\Delta_0$ formulas, by the Closure Lemma 2.4.6, D must also be an $\exists\Delta_0$ formula. So by Corollary 2.4.5.1 and the soundness assumption of $\mathcal{T}$ (as stated in Section 1.3), we have:

$$\mathcal{T} \vdash D \iff D \text{ is true}$$

For the final observation, notice that $\varphi(\overline{k}, \overline{k}, y)$ is true iff $sub(\ulcorner B(x) \urcorner, k) = y$, which is true iff $y = \ulcorner B(\overline{k}) \urcorner$, so combining everything altogether:

$$\begin{aligned} \mathcal{T} \vdash D &\iff D \text{ is true} \\ &\iff \exists y(A(y) \wedge \varphi(\overline{k}, \overline{k}, y)) \\ &\iff \exists y(A(y) \wedge y = \ulcorner B(\overline{k}) \urcorner) \\ &\iff A(\ulcorner B(\overline{k}) \urcorner) \\ &\iff A(\ulcorner D \urcorner) \end{aligned}$$

□

4.2.2 The Main Proof

Finally, we tackle the main proof, recall the theorem:

Theorem 4.2.2. ***Gödel's First Incompleteness Theorem***
Every consistent formal system containing $\mathbf{Q}$ *cannot be complete.*

Proof. Suppose we have consistent formal system $\mathcal{T}$ which contains $\mathbf{Q}$. Also suppose for the sake of contradiction, that $\mathcal{T}$ is complete. Consider the binary relation:

$$Prf(x, y) : x \text{ is the proof of } y$$

where x is the super Gödel number of a proof, and y is the Gödel number of a sentence. Using this relation, we can construct a new relation: "the sentence encoded by Gödel number y is provable":

$$Prov(y) : \exists x Prf(x, y)$$

Notice that the negated provability statement: $\neg Prov(y)$, is semi-decidable. To check, suppose y decodes into sentence A, by completeness and consistency, either $\neg A$ or A (not both) must be a theorem of $\mathcal{T}$, so we only need to find $\neg A$ as a theorem of $\mathcal{T}$ to show A is not provable. Since the property of being a theorem is semi-decidable (by effective axiomatization), $\neg Prov(y)$ is also semi-decidable.

Then by the Representability Theorem 2.3.1, there is some $\exists\Delta_0$ formula $\varphi(y)$ which weakly represents the relation $\neg Prov(y)$. Using this formula,

employ the Diagonal Lemma 4.2.1, we are able to construct sentence G such that:

$$\begin{aligned} \mathcal{T} \vdash G &\iff \varphi(\ulcorner G \urcorner) \\ &\iff \neg Prov(\ulcorner G \urcorner) \end{aligned}$$

The above line reads: **"$\mathcal{T}$ proves G iff G is not provable"**, and now the question is, can we manage to prove, or disprove G?

- If $\mathcal{T}$ proves G, then that implies G is provable, meanwhile the "iff" tells us that G is not provable, hence $\mathcal{T}$ is not consistent, contradiction!
- If $\mathcal{T}$ disproves G, that is, it proves $\neg G$, then it implies G is not provable (by consistency assumption). But the "iff" tells us that G is provable, again contradicting consistency.

Therefore, G is a sentence which we cannot prove nor disprove, contradicting our previous assumption of completeness, hence the conclusion that the formal system $\mathcal{T}$ is indeed, incomplete. We are done! □

Epilogue

This concludes two proofs for Gödel's First Incompleteness Theorem, I really hope this text can help someone learn something new.

Finally, many thanks to Professor Todorcevic at UofT who has made all of this possible. It was an absolute joy composing this write-up. There are also works that I just can't appreciate enough:

- *Gödel Without (Too Many) Tears* by Professor Smith
- *Stanford Encyclopedia of Philosophy* on Gödel's Incompleteness Theorems by Professor Raatikainen
- Professor Cook's notes for *CSC438: Computability and Logic*, during his teaching here at UofT!

These are incredibly clear and well-structured works, and I learned all of the proof ideas in this text from them. If you are interested, make sure to check them out, cheers!