Mathematical Limits

Logic and Computational Theorems

Amir H. Ebrahimnezhad

June 8, 2023

Contents

1 Ground Works

1.1 Vorwort

Gödel's incompleteness theorems, are two theorems of mathematical logic that deal with the limits of provability in formal axiomatic theories. The theorems are interpreted as showing that Hilber's program to find a complete and consistent set of axioms for all mathematics is impossible.

In the following work we will first describe the theorem and preliminaries, and then prove it using algorithm theory methods and other methods. Then we are going to talk about it's consequences in pure mathematics and philosophy of mathematics, later we will discuss the relation between the theorems and the church-turing thesis We then would discuss related works that has pushed the theorem further and shown aspects of it. At the final section we will investigate a philosophical picture of Mathematical Universe and if it is acceptable to consider a bigger abstract world of mathematical objects where the real world is a subset of it.

2 Formal Language Theory

The investigation of Incompleteness theorem requiers the knowledge of some basic ideas such as languages, grammars, automatas and others. Therefore we start this draft by talking about these topics in a general format. A more detailed version would be published later.

2.1 Languages

We would begin by the definition of a language. In a informal way a language is what we speak or what we write on a paper to give information to another person, mathematically and a more abstract idea of a language is a bunch of symbols that we have to make words (gluing symbols together to resemble a unity of them); To give a precise mathematical definition of language we would first define alphabet as:

> ***Definition*** 1 –*An alphabet Σ, is a set of symbols.*

> ***Definition*** 2 –*A word M, is a combination of symbols from alphabet Σ.*

For instance one could choose the set $\{a, b, c, d, \ldots\}$ as ones alphabet. Then "cat" is a word in it's alphabet. As you can see there's a meaning for the word we gave as an example. It defines an object (more accurately a living creature). Since there can be Infinetly many words for any alphabet (other than an empty set which have no words), we distinguish between the accepted combinations and the ones we don't accept by defining a grammar for our language; A grammar is a way to characterize a language, a way to list which stings of Σ is acceptable. We could simply list strings or have a set of rules (or an algorithm) to say if a given combination is acceptable or not for a language. Thus we define a language as:

> ***Definition*** 3 –*Given an alphabet Σ, Σ^∞ is the set of all possible words in the alphabet.*

> ***Definition*** 4 –*A subset S of a set X is decidable if and only if there exists a function that given $x \in X$ decides if $x \in S$ is true or false.*

> ***Definition*** 5 –*A Language L, is a subset of the alphabet Σ^∞ ($L \subset \Sigma^\infty$) where there exists a function $\eta(\sigma \in \Sigma^\infty)$ called grammar that decides L.*

Formally, we define a grammar as:

> ***Definition*** 6 –*A Grammar is a set $\{V_T, V_N, S, R\}$ where V_T is the set of terminal elements, V_N is the set of non-terminal elements, S is a memeber of V_N, and R is a finite set of rules.*

We would use these definitions in the later sections of this draft. But for now let us give a formal definition of R as well:

> ***Definition*** 7 –*R is a finite set of ordered pairs from $\Sigma^\infty V_N \Sigma^\infty \times \Sigma^\infty$, where $\Sigma = V_T \cup V_N$.*

In later drafts we would get back to the Formal language theory in more depth and examine the properties of the setsof languages each grammar formalism can accomodate and the set of abstract machines that correspond to each type. But for now we would start with the things we need for proving the incompleteness theorems that is the goal of this draft.

§ **Induction:** Induction is a method of proving that a statemens $P(n)$ is true for every natural number n, that is, that the infinitely many cases, $P(0), P(1), \ldots$ all hold. [1]

> "Mathematical induction proves that we can climb as high as we like on a ladder, by proving that we can climb onto the bottom rung (the basis) and that from each rung we can climb up to the next one (the step)."

Theorem 1 — For every natural number n,

$$1+2+\cdots+n=\frac{n(n+1)}{2} \tag{1}$$

► *proof:* If $n=1$, the equality holds. For the inductive case, fix $k \geq 1$ and assume that:

$$1+2+\cdots+k=\frac{k(k+1)}{2} \tag{2}$$

Now adding $k+1$ to each side we have:

$$1+2+\cdots+(k+1)=\frac{k(k+1)}{2}+(k+1) \tag{3}$$

Since the right hand side simplifies to:

$$\frac{(k+1)((k+1)+1)}{2} \tag{4}$$

Finishing the inductive step and thus the proof. As you can see in the inductive step what we prove is that:

> "*If the formula holds for k, then the formula holds for $k+1$.*"

Looking at this from a slightly different angle, what we have done is to construct a set of numbers with a certain property. If we let S stand for the set of numbers for which our theorem holds, in out proof by induction we whow that the set S, is identical with the set of natural numbers, thus the theorem holds for every natural number n, as needed.

So what makes a proof by induction work is the fact that the natural numbers can be defined recursively, There is a base case, consisting of the smallest natural number, and there is a recursive case, showing how to construct bigger natural number from smaller ones.

§ **Terms and Formulas:** As we mentioned earlier not all words of a set Σ^{∞} is meaningful. Since any combination of the alphabet is a word there has to be distinctions between what are meaningful words and what are not. We would consider two kinds of words as *terms & formulas* as follow:

> ***Definition*** 8 –*If $\mathscr{L}$ is a language, a* ***term of*** *$\mathscr{L}$ is a nonempty finite string t of symbols from $\mathscr{L}$ such that either:*
>
> 1. t is a variable, or
> 2. t is a constant symbol, or
> 3. $t :\equiv f t_1 t_2 t_3 \dots t_n$, where f is an n-ary function symbol of $\mathscr{L}$ and each of the t_i is a term of $\mathscr{L}$.

> ***Definition*** 9 –*If $\mathscr{L}$ is a language, a formula of $\mathscr{L}$ is a nonempty finite string of ϕ of symbols from $\mathscr{L}$ such that either:*
>
> 1. $\phi :\equiv = t_1 t_2$, where t_1, t_2 are terms of $\mathscr{L}$, or
> 2. $\phi :\equiv R t_1 t_2 \dots t_n$ where R is an n-ary relation symbol of $\mathscr{L}$ and $t_1, t_2, \dots, t_n$ are all terms of $\mathscr{L}$, or
> 3. $\phi :\equiv (\neg\alpha)$ where α is a formula of $\mathscr{L}$, or
> 4. $\phi :\equiv (\alpha \vee \beta)$, where α and β are formulas of $\mathscr{L}$, or
> 5. $\phi :\equiv (\forall v)(\alpha)$, where v is a variable and α is a formula of $\mathscr{L}$

Notice that the five clauses of the definition can be separated into two groups. The first two clauses, the atomic formulas, are explicitly defined. The last three clauses are the recursive case, showing how if α and β are formulas, they can be used to build more complex formulas, such as $(\alpha \vee \beta)$ or $(\forall v)(\alpha)$. Now since the collection of formulas is defined recursively, we can use an inductive style proof when we want to prove that something is true about every formula. The inductive proof will consist of two parts, a base cae and an inductive case. First we prove the statement for every atomic formula and then using the inductive method we prove it for recursive formulas from the atomic ones. *This method is called induction on the complexity of the formula, or induction on the structure of the formula.*

§ **A First-order Language:** Before getting to Sentences one should know a definition for a first-order language. A first-order language $\mathscr{L}$ is defined as an infinite collection of symbols, separated into the following categories:

- *Parentheses:* $(,)$.
- *Connectives:* $\wedge, \vee, \neg$.
- *Quantifier:* $\forall, \exists$.
- *Variables:* one for each positive integer n denoted: v_n for nth number.
- *Equality:* $=$.
- *Constant:* We can have a new symbol for each positive number or any other method that we distinguish between two numbert (we can use | for 1, || for 2 etc)
- *Functions:* For each positive integer n, some set of zero or more n-ary function symbols.
- *Relation:* For each positive integer n, some set of zero or more n-ary relation symbols.

Callout — Having n arity means that it is intended to represent a function of n variables.

Notice that by defining such language one can avoid the process of finiding an algorithm of grammar (the one that differenciates between nonesense and meaningful words) since we defined all the possible functions and etc. This way we have to only

§ **Sentences:** Among the formulas in the language $\mathscr{L}$, there are some in which we will be especially interested. These are the sentences of $\mathscr{L}$. The formulas that can be either true or false in a given mathematical model.

> ***Definition*** 10 –*A sentence in a language $\mathscr{L}$ is a formula of $\mathscr{L}$ that contains no free variable.*

§ **Structures:** Defining any language and with constansts, functions and etc. There can exist multiple ways to define structures as opposed to another. The point is that there is no preference. Thus without determining the structure under consideration, without deciding how we wishs to interpret the symbols of the language, we have no way of talking about the truth or falsity of a sentence. Thus we have:

> ***Definition*** 11 –*Fix a language $\mathscr{L}$. An $\mathscr{L}$-Structure $\mathscr{A}$ is a nonempty set Am caled the **Universe of** $\mathscr{A}$, together with:*
>
> 1. *For each constant symbol c of $\mathscr{L}$, an element $c^{\mathscr{A}}$ of A*
> 2. *For each n-ary function symbol f of $\mathscr{L}$, a function $f^{\mathscr{A}} LA^n \to A$, and*
> 3. *For each n-ary relation symbol R of $\mathscr{L}$, and n-ary relation $R^{\mathscr{A}}$ on A.*

§ **Truth in a Structure:** Now that we know some formal rules about what constitutes a language, we would like to merge syntax and semantics. We want to answer what is means to say that an $\mathscr{L}$-formula is true in an $\mathscr{L}$-structure $\mathscr{A}$.

To begin the process of tying together the symbols with the structures, we will introduce assignment functions. These assignment functions will formalize what it means to interpret a term or a formula in a structure.

> ***Definition*** 12 –*if $\mathscr{A}$ us an $\mathscr{L}$-structure, a **variable assignment function into** $\mathscr{A}$ is a function s that assigns to each variable an element of the universe A. So a variable assignment function into $\mathscr{A}$ is any function with domain V and codomain A.*

We will have occasion to want to fix the value of the assignment function s for certain variables, then:

> ***Definition*** 13 –*If s is a variable assignment function into $\mathscr{A}$ and x is a variable and $a \in A$, then $s[x|a]$ is the variable assignment function into $\mathscr{A}$ defined as follows:*
>
> $$s[x|a](v) = \begin{cases} s(v) & \text{if } v \text{ is a variable other than } x \\ a & \text{if } v \text{ is the variable} x \end{cases} \tag{5}$$

We call the function $s[x|a]$ and x-modification of the assignment function s. This is essentially the same function, except that the variable x is now assigned to a particular element of the universe.

What we will do next is extend a variable assignment function to a term assignment function $\bar{s}$. Thius function will assign an element of the universe to each term of the language $\mathscr{L}$.

> ***Definition*** 14 –*Suppose that $\mathscr{A}$ is an $\mathscr{L}$-structure and s is a variable assignment function into $\mathscr{A}$. The function $\bar{s}$ is called the term assignment function generated by s, is the function with domain consisting of the set of $\mathscr{L}$-terms and codomain A defined recursively as follows:*
>
> 1. *If t is a variable, $\bar{s}(t) = s(t)$.*
> 2. *If t is a constant symbol c, then $\bar{s}(t) = c^{\mathscr{A}}$.*
> 3. *If $t :\equiv f t_1 t_2 \ldots t_n$, then $\bar{s}(t) = f^{\mathscr{A}}(\bar{s}(t_1), \bar{s}(t_2), \ldots, \bar{s}(t_n))$.*

Although we will be primarily interested in truth of sentences, we will first describe truth (or satisfaction) for arbitrary formulas, relative to an assignment function.

> ***Definition*** 15 –*Suppose that $\mathscr{A}$ us an $\mathscr{L}$-structure, ϕ is an $\mathscr{L}$-formula, and $s\colon V \to A$ is an assignment function. We will say that $\mathscr{A}$ satisfies ϕ with assignment s, and write $\mathscr{A} \vDash \phi[s]$, in the following circumstances:*
>
> 1. *If $\phi :\equiv = t_1 t_2$ and $\bar{s}(t_1)$ is the same element of the universe A as $\bar{s}(t_2)$, or*
> 2. *If $\phi :\equiv R t_1 \ldots t_n$ and $(\bar{s}(t_1), \ldots, \bar{s}(t_n)) \in R^{\mathscr{A}}$, or*
> 3. *If $\phi :\equiv (\neg\alpha)$ and $\mathscr{A} \nvDash \alpha[s]$ (where $\nvDash$ means "does not satisfy") or*
> 4. *If $\phi :\equiv (\alpha \vee \beta)$ and $\mathscr{A} \vDash \alpha[s]$, or $\mathscr{A} \vDash \beta[s]$ (or both), or*
> 5. *$\phi :\equiv (\forall x)(\alpha)$ and, for each element a of A, $\mathscr{A} \vDash \alpha[s(x|a)]$*

Callout — If Γ is a set of $\mathscr{L}$-formulas, we say that $\mathscr{A}$ satisfies Γ with assignment s, and write $\mathscr{A} \vDash \Gamma[s]$ if for each $\gamma \in \Gamma, \mathscr{A} \vDash \gamma[s]$

§ **Substitutions and Substitutability:** Suppose that you knew the sentence $\forall x \phi(x)$ was tru in particular structure $\mathscr{A}$. Then, if c is a constant symbol in the language, you would certainly expect $\phi(c)$ to be true in $\mathscr{A}$ as well.

Suppose now that $\mathscr{A} \vDash \forall x \exists y \neg(x = y)$. This sentence is, in fact, true in any structure $\mathscr{A}$ such that A has atleast two elements. The rules of substitutability that we will discuss in this section are designed to help us avoid this problem, the problem of attempting to substitute a term inside a quantifier that binds a variable involved in the term.

> ***Definition*** 16 –*Suppose that u is a term, x is a variable, and t is a term. We define the term u^x_t, and read u with x replaced by t as:*
>
> 1. *If u is a variable not equal to x, then u^x_t is u*
> 2. *If u is x, then u^x_t is t*
> 3. *If u is a constant symbol then u^x_t is u*
> 4. *If $u :\equiv f u_1 u_2 \ldots u_n$, where f is a n-ary function symbol and the u_i are terms, then:*
>
> $$u^x_t \text{ is } f(u_1)^x_t \ldots (u_n)^x_t \tag{6}$$

Having defined what it means to substitute a term for a variable now we define what is substitutability:

Definition 17 –*Suppose that ϕ is a $\mathscr{L}$-formula, t is a term, and x is a variable. We say that t is substitutable for x in ϕ if:*

1. *ϕ is atomic, or*
2. *$\phi :\equiv \neg(\alpha)$ and t is substitutable for x in α, or*
3. *$\phi :\equiv (\alpha \vee \beta)$ and t is substitutable for x in both α and β*
4. *$\phi :\equiv (\forall y)(\alpha)$ and either*
   (a) *x is not free in ϕ, or*
   (b) *y does not occur in t andt is substitable or x in α.*

§ **Logical Implication:** In this section we formalize the question that If I know this statement is true, is it necessarily the case that this other statement is true.

Definition 18 –*Suppose that Δ and Γ are sets of $\mathscr{L}$-formulas. We will say that Δ logically implies Γ and write $\Delta \models \Gamma$ if for every $\mathscr{L}$-structure $\mathscr{A}$, if $\mathscr{A} \models \Delta$, then $\mathscr{A} \models \Gamma$.*

This definition says that if Δ is true in $\mathscr{A}$, then Γ is true in $\mathscr{A}$. Remember, for Δ to be true in $\mathscr{A}$, it must be the case that $\mathscr{A} \models \Delta[s]$ for every assignment function s.

Definition 19 –An $\mathscr{L}$-formula ϕ is said to be valid if $\emptyset \models \phi$, in other words, if ϕ is true in every $\mathscr{L}$-structure with every assignment function s. In this case we will write $\models \phi$.

Callout — For the double turnstyle symbol $\models$, if there is a structure on the left, $\mathscr{A} \models \sigma$, we are discussing truth in a single structure. On the other hand if there is a set of sentences on the left $\Gamma \models \sigma$, then we are discussing logical implication.

2.2 Deductions

In this section we will try to formalize what is known to be the deductive method. In mathematics we generally tend to insist upon the existence of a proof for any true statement (or at least we hope so). A proof is a sequence of statements, each one of which can be justified by referring to previous statements. This is a perfectly reasonable starting point, and it brings us to the main difficuly we will have to address as we move from an informal understanding of what constitutes a proof to a formal definition of deduction.

The proofs that you have seen in your mathematical career have had a couple of nice properties. The first of these is that proofs are easy to follow. This doesn't mean that it is easy to discover a proof, but rather that if someone is showing you a proof, it should be easy to follow the steps of the proof and to understand why the proof is correct. The second admirable property of proofs is that when you prove something, you know that it is true! Our definition of deduction will be designed to make sure that deductions, too, will be easily checkable and will preserve truth.

We then would impose the following restrictions on our logical axioms and rules of inference:

1. There will be an algorithm that will decide, given a formula, whether or not that formula is a logical axiom.
2. There will be an algorithm that will decide, given a finite set of formulas Γ and a fimula θ, whether or not (Γ, θ) is a rule of inference.
3. For each rule of inference (Γ, θ), Γ will be a finite set of formulas.
4. Each logical axiom will be valid.
5. Our rules of inference will preserve truth. In other words, for each rule of inference $(\Gamma, \theta) \to \Gamma \models \theta$.

The idea is that there should be no brilliance and no insight required to check whether an alleged deduction of α is, infact, a deduction of α. To check whether a deduction is correct will be such a simple procedure that it could be programmed into a computer.

We begin by fixing a language $\mathscr{L}$. Also assume that we have been given a fixed set of $\mathscr{L}$-formulas, Λ, is called the set of logical axioms, and a set of ordered pairs (Γ, ϕ), called the rules of inference. A deduction is going to be a finite sequence or ordered list, of $\mathscr{L}$-formulas with certain properties.

> ***Definition*** 20 –*Suppose that* Σ *is a collection of* $\mathscr{L}$*-formulas and* D *is a finite sequence* $(\phi_1, \ldots, \phi_n)$ *of* $\mathscr{L}$*-formulas. We will say that* D *is a deduction form* Σ *if for each element in* D*:*
>
> 1. $\phi_i \in \Lambda$ (ϕ_i is a logical axiom), or
> 2. $\phi_i \in \Sigma$ (ϕ_i is a nonlogical axiom), or
> 3. There is a rule of inference (Γ, ϕ_i) such that $\Gamma \subseteq \{\phi_1, \ldots, \phi_{i-1}\}$

If there is a deduction form Σ, the last line of which is the formula ϕ, we will call this a deduction form Σ of ϕ. And write: $\Sigma \vdash \phi$.

Callout — Well, we have now established what we mean by the word justified. In a deduction we are allowed to write down any $\mathscr{L}$-formula that we like, as long as that formula is either a logical axiom or is listed explicitly in a collection Σ of nonlogical axioms. Any formula that we write in a deduction that is not an axiom must arise from previous formulas in the deduction via a rule of inference.

2.2.1 Logical Axioms

In this section we will gather together a collection of Λ of logical axioms for $\mathscr{L}$. This set of axioms, though infinite, will be decidable. Which means that there exists an algorithm, which given an input x, can tell if $x \in \Lambda$ is true or false.

§ **Equality Axioms:** We have taken the route of assuming that the equality symbol, =, is a part of the language iteself. There are three groups of axioms that are designed for this symbol. The first just says that any object is equal to itself:

$$x = x \tag{7}$$

For the second group of axioms, assume that x_i and y_i are variables, and f is an n-ary function symbol.

$$\big[(x_1 = y_1) \wedge (x_2 = y_2) \wedge \cdots \wedge (x_n = y_n)\big] \rightarrow \big(f(x_1, x_2, \ldots, x_n) = f(y_1, y_2, \ldots, y_n)\big) \tag{8}$$

The assumption for the third group of axioms is the same as for the second group, except that R is assumed to be an n-ary relation symbol.

$$\big[(x_1 = y_1) \wedge (x_2 = y_2) \wedge \cdots \wedge (x_n = y_n)\big] \rightarrow \big(R(x_1, x_2, \ldots, x_n) = R(y_1, y_2, \ldots, y_n)\big) \tag{9}$$

§ **Quantifier Axioms:** The quantifier axioms are designed to allow a very reasonable sort of entry in a deduction. Suppose that we know $\forall x P(x)$. Then, if t is any term of the language, we should be able to state that $P(t)$. To avoid some problems we will demand that the term t be substitutable for the variable x.

$$(\forall x \phi) \rightarrow \phi^x_t, \text{if } t \text{ is substitutable for } x \text{ in } \phi \tag{10}$$

$$\phi^x_t \rightarrow (\exists x \phi), \text{if } t \text{ is substitutable for } x \text{ in } \phi \tag{11}$$

In many logic texts, the first axiom would be called universal instantiation, while the second would be known as existential generalization.

2.2.2 Rules of Inference:

There will be two types of Rules of Inference, one dealing with propositional consequence and one dealing with quantifier.

§ **Propositional Consequence:**We work with a restricted language $curveP$, consisting only of a set of propositional variables $A, B, C, \ldots$ and the connectives $\vee$ and $\neg$. Notice there are no quantifiers, no relation symbols, no function symbols, and no constansts. Each propositional variable can be assigned one of two truth values, T,F for truth and falsity respectively. The we can define a function v to assign the truth value and for extending that we would define:

$$\bar{v}(\phi) = \begin{cases} v(\phi) & \text{if } \phi \text{ is a propositional variable} \\ F & \text{if } \phi :\equiv (\neg\alpha) \text{ and } \bar{v}\alpha = T \\ F & \text{if } \phi :\equiv (\alpha \vee \beta) \text{and } \bar{v}(\alpha) = \bar{(}\beta) = F \\ T & \text{otherwise} \end{cases} \tag{12}$$

To discuss propositional consequence in first-order logic, we will transfer our formulas to the real of propositional logic and use the idea of tautology in that area. To be specific, given β , an $\mathscr{L}$-formula of first-order logic, gere is a procedure that will convert β to a formula β_P of propositional logic corresponding to β:

1. Find all subfomulas of β of the form $\forall x \alpha$ that are not in the scope of another quantifier. Replace them with propositional variables in a systematic fashion. This means that if $\forall y Q(y,c)$ appears twice in β it is replaced by the same letter both times, and distinct subformulas are replaced with distict letters.
2. Find all aromic formulas that remain, and replace them systemically with new propositional variables.
3. At this point, β will have been replaced with a propositional formula β_P

We are now almost at a point where we can state out propositional rule of inference. Recall that a rule of inference is an ordered pair (Γ, ϕ), where Γ is a set of $\mathscr{L}$-formulas and ϕ is a $\mathscr{L}$-formula.

> ***Definition*** 21 –*Suppose that Γ_P is a set of propositional formulas and ϕ_P is a propositional formula. We will say that ϕ_P is a propositional consequence of Γ_P if every truth assignment that makes each propositional formula in Γ_P true also makes ϕ_P true. Notice that ϕ_P is a tautology if and only if ϕ_P is a propositional consequence of $\emptyset$.*

Notice that if $\Gamma_P = \{\gamma_{1P}, \ldots, \gamma_{nP}\}$ is a nonempty finite set of propositional formulas and ϕ_P is a propositional formula, then ϕ_P is a propositional consequence of Γ_P if and only if:

$$[\gamma_{1P} \wedge \cdots \wedge \gamma_{nP}] \to \phi_P \tag{13}$$

is a tautology.

> Callout — A tautology is a formula of assertion that is true in every possible structure $\mathscr{A}$.

Now we extend our definition:

> ***Definition*** 22 –Suppose that Γ is a finite set of $\mathscr{L}$-formulas and ϕ is an $\mathscr{L}$-formula. We will say that ϕ is a propositional consequence of Γ if ϕ_P is a propositional consequence of Γ_P, where the two latter are the results of applying the procedure of making a first order logic formulas into propositional formulas.

> ***Definition*** 23 –If Γ is a finite set of $\mathscr{L}$-formulas, ϕ is an $\mathscr{L}$-formula, and ϕ is a propositional consequence of Γ then (Γ, ϕ) is a rule of inference of type (PC).

§ **Quantifier Rules** Suppose, without making any particular assumptions about x, that you were able to prove x is a then you also have proved $(\forall x) x$ is a. Looking at it from back to front we would have:

> ***Definition*** 24 –Suppose that the variable x is not free in the formula ψ. Then both of the following are rules of inference of type (QR).
>
> $$\left(\{\psi \to \phi\}, \psi \to (\forall x \phi)\right) \tag{14}$$
> $$\left(\{\phi \to \psi\}, \exists x \phi \to \psi\right) \tag{15}$$

2.3 Soundness

Generally in mathematics we would like to be sure that when something has been proved, then it is for sure, true. In thi section we will prove a theorem that shows that the logical system that we have developed has this highly desirable property. This result is called the Soundness Theorem. As you might recall the requirements that we set for our rule of inference are:

1. There will be an algorithm that will decide, given a formula θ, whether or not θ is a logical axiom.
2. There will be an algorithm that will decide, given a finite set of formulas Γ and a formula θ, whether or not (Γ, θ) is a rule of inference.
3. For each rule of inference (Γ, θ), Γ will be a finite set of formulas.
4. Each logical axiom will be valid.
5. Our rules of inference will preserve truth, or in other words, for each rule of inference (Γ, θ), $\Gamma \vDash \theta$.

These requirements serve two purposes: They allow us to verify mechanically that an alleged deduction is in fact a deduction, and they provide the basis of the Soundness Theorem. The idea behind the Soundness theorem is very simple. Suppose that Σ is a set of $\mathscr{L}$-formulas and suppose that there is a deduction of ϕ from Σ. What the Soundness Theorem tells us is that in any structure $\mathscr{A}$ that makes all of the formulas of Σ true, ϕ is true as well.

Theorem 2 — If $\Sigma \vdash \phi$, then $\Sigma \vDash \phi$

proof. Let $\text{Thm}_\Sigma = \{\phi | \Sigma \vdash \phi\}$ and $C = \{\phi | \Sigma \vDash \phi\}$. By showing that $\text{Thm} \subseteq C$ we prove the theorem. Notice that C has the following characteristics:

1. $\Sigma \subseteq C$. If $\sigma \in \Sigma$ then certainly $\Sigma \vDash \sigma$.
2. $\Lambda \subseteq C$. As the logical axioms are valid, they are true in any structure. Thus $\Sigma \vDash \lambda$ for any logical axiom.
3. If (Γ, θ) is a rule of inference and $\Gamma \subseteq C$, then $\theta \in C$. Then because: $\mathscr{A} \vDash \Gamma$ and $\Gamma \vDash \theta$ we know that $\mathscr{A} \vDash \theta$.

Since we have the following proposition and C has these characteritics the prove is completed: *Fix sets of $\mathscr{L}$-formulas Σ and Λ and a collection of rules of inference. The set Thm_Σ is the smallest set C such:*

- $\Sigma \subseteq C$
- $\Lambda \subseteq C$
- If (Γ, θ) is a rule of inference and $\Gamma \subseteq C$, then $\theta \in C$.

3 Completeness

Before beginning with the incompleteness theorem it might help to know actually what is completeness. We have established a deductive system consisting of logical axioms and rules of inference. The Soundness Theorem showed that our deductive system preserves truth; that is, if there is a deduction of a formula φ from a set of formulas Σ, then φ is true in any model of Σ. Formally, we write this as follows:

$$\Sigma \vdash \varphi \implies \Sigma \models \varphi$$

where $\vdash$ denotes provability in the deductive system, and $\models$ denotes semantic entailment.

The Completeness Theorem is the first major result of this chapter, and it gives us the converse to the Soundness Theorem. Specifically, it states that every semantically valid formula is provable in our deductive system. In other words, if a formula is true in every model, then it can be proved using only the logical axioms and rules of inference we have established. Formally, we write this as follows:

$$\Sigma \models \varphi \implies \Sigma \vdash \varphi \tag{16}$$

Combined with the Soundness Theorem, the Completeness Theorem gives us the following equivalence:

> ***Definition*** 25 –A deductive system consisting of a collection of logical axioms Λ and a collection of rules of inference is said to be complete if for every set of nonlogical axioms Σ and every $\mathscr{L}$-formula ϕ,
>
> $$\text{If } \Sigma \vDash \phi, \text{ then } \Sigma \vdash \phi \tag{17}$$

This equivalence assures us that if our deductive system allows us to prove a statement, then that statement is true, and conversely, if a statement is true in every model, then it can be proved using our deductive system.

However, it is important to note that the Completeness Theorem only applies to formulas that are true in every possible model. It does not necessarily extend to formulas that are true in some but not all models. Furthermore, the theorem applies specifically to first-order logic and may not hold for other logical systems.

4 Incompleteness Theorems

4.1 Mathematics

4.1.1 Prespective

In the last section we talked about the completeness theorem, which investigated the fact that our deductive system is sound and complete. For the collection of lpgical axioms that we have set out, any formula ϕ that can be deduced from a set of nonlogical axioms Σ will be true in all models of Σ under any variable assignment function (soundness), and furthermore any formula ϕ that is true in all models of Σ under every assignment functionwill be deducible from Σ (completeness)–This means that we can prove every true statement within the system. Thus our deductive system is the best it can be.

Now in this section we would try to show how would Incompleteness be proved, since we have shown that th efirst order logic is sound and complete it is trvial to infere that Incompleteness and Completeness are to be proved for different languages, and wether or not we can use them in Physics is to be worked out separately. But in the last sections we surely developed necessary concepts for one, who would like to talk about completeness (or incompleteness) of a physical theory.

Later on we would dicuss why in uor view. This might be the case.

4.1.2 Basic Theory of Algorithm

We assume that we are given a subset T of language L, which is called the set of *true statements*. This set should contain only the statements in the language that would evaluate as True. In the process of assuming such subset we ommit the part where we would consider the statemens true or false. A language then would be completely defined (for our purpose) if we are given the *fundamental pair:*

> ***Definition*** 26 –*Given a language L and the subset of true statements T, we call $\langle L, T\rangle$ a fundamental pair.*

> ***Definition*** 27 –***Unprovable*** *means not provable, and provable means there exist a proof for such statement.*

Also the concept of proof, deduction, and induction were defined in previous sections. For the sake of simplicity consider a deductive system that consists of an alphabet called *Alphabet of Proofs*, and denoted P. Then we would generally have P^∞ as all the words that alphabet can have, but restrictions by grammar, rules of inference, induction, deduction etc, would provide us with a set of proofs called $\bar{P}$. As you would recall our deductive system would prove a statement, and only one for each proof. Therefore given a language $\mathscr{L}_x$ and a proof language P one would require the existence of an algorithm that can map a proof from P to a true statement from $\mathscr{L}_x$. Therefore a deductive system (in the most simple manner, but you can still consider the deduction introduced in the previous sections.) would be made of P, $\bar{P}$, δ, where δ is a function that maps proofs from P to true statements from $\mathscr{L}_x$.

> ***Definition*** 28 –*We would call $\langle P, \bar{P}, \delta\rangle adeductivesystem$.*

So our final definition is as follows:

1. We have a language $\mathscr{L}$ and an alphabet P for the proofs.
2. In the set P^∞ we are given a subset $\bar{P}$, whose elements are called proofs. We futher assume:
   (a) We shall allow different concepts of proofs, different proof subsets of P^∞. We shall also allow the alphabet P to vary.
   (b) There has to be an effective method or algorithm to check if a given word in P is a proof and to show what stetement it proves.
3. We have a function δ (to determine what is being proved) whose domain of definition Δ satisfies $\bar{P} \subseteq \Delta \subseteq P^\infty$, and whose range of valurs is in L^∞. We assue that we have an algorithm which computes this function. Therefore we might say that the proof p in $\bar{P}$ is the proof of $\delta(p)$ in the language $\mathscr{L}$.

Therefore we can now define consistency and completeness more easily:

> ***Definition*** 29 –*We say the deductive system $\langle P, \bar{P}, \delta\rangle$ is consistent relative to the fundamental pair $\langle \mathscr{L}, T\rangle$ if we have $\delta(\bar{P}) \subseteq T$. In other words our deductive system would not prove contradictory statements.*

> ***Definition*** 30 –*We say the deductive system $\langle P, \bar{P}, \delta\rangle$ is complete relative to the fundamental pair $\langle \mathscr{L}, T\rangle$ if we have $T \subseteq \delta(\bar{P})$. In other words our deductive system would prove every true statements within the language.*

Conditions for the nonexistence of a complete and consistent deductive system can easily be given in terms of the theory of algorithms. An algorithm is a set of instructions which, given an input enables us to obtain an output if such an output exists or else obtain nothing at all if theres no output for out particular input. For our purpose consider that every input and output is a word thus there exists alphabet I for inputs and alphabet O for outputs.

In other words, in order to work with algorithms which process paris or sequences of words we must first compress such pairs or sequences of as single words in the algorithm, in a new alphabet.

Let $\star$ denote for a new letter not in our alphabet $\mathscr{L}$. Thus $\mathscr{L}_\star$ is a new alphabet. We would use such letter to denote separation between words in our alphabet.

Now we must define some terms:

> ***Definition*** 31 –***Domain of Applicability:*** *The set of all inputa that can be processes by a given algorithm is called the domain of applicability of the algorithm.*

> ***Definition*** 32 –*The algorithm A computes the function f, if we have $A(x) \sim f(x)$ for all x*

Callout — Note that $\sim$ is the conditional equality sign, both sides are equal when they are defined and bot sides are undefined under same x.

> ***Definition*** 33 –*A function that can be computed by some algorithm is called a computable function.*

> ***Definition*** 34 –***Decidable:*** *A subset S of a set A is said to be decidable if there exsits and algorithm which determines whether or not an element of A is in S.*

We would require the set of all proofs be decidable set of all the words in proof alphabet.

Lemma 1 –For any subset X, the set $\emptyset$ and X are decidable relative to X.

► *proof:* We provide an algorithm for the subset X of X that always returnes true. And for the $\emptyset$ always false.

Theorem 3 — If T is an enumerable set, then one can find a complete and consistent deductive system for the fundamental pair $\langle \mathscr{L}, T\rangle$

► *proof:* if $T = \emptyset$ the deductive system $\langle P, \bar{P}, \delta\rangle$ where $\bar{P} = \emptyset$ and $delta(p) = \emptyset$ is consistent and complete relative to T.
Otherwise if $T \neq \emptyset$ since it is enumerable (there exists an algorithm that maps the natural numbers to the set.), we would have the algoritm τ that enumerates T. We would later prove that the set of all proofs is enumerable therefore by choosing $\delta = \tau$ we would have a consistent and complete deductive system.

Lemma 2 –A decidable subset of an enumerable set is enumerable.

► *proof:* if f is the enumerating function of a set A. If $S \subseteq A$ is empty set. Then it is enumerable by definition. If not there exists $s \in S$ thus we enumerate the set by the following function:

$$g(n) = \begin{cases} f(n) & \text{if } f(n) \in S \\ s & \text{otherwise} \end{cases} \tag{18}$$

Lemma 3 –A subset S of an enumerable set X is decidable relative to X is decidable relative to X if and only if both S and its complement $X \backslash S$ are enumerable.

► *proof:* If S is decidable, then so is $X \backslash S$, and so both S and $X \backslash S$ are enumerable by Lemma 2. Conversely, suppose that both S and $X \backslash S$ are enumerable. If either one is empty then, S is decidable, by Lemma 1. Suppose both S and its complement are nonempty sets, in which case they are enumerated by some computable functions f and g, respectively. Then, for deciding x we need only compute $f(0), g(0), f(1), g(1), \ldots$ until we encounter x.

Theorem 4 — The set of all proofs is enumerable.

► *proof:* Fir any enumerable alphabet L we have:

$$L^{\infty} = \{x|x \in L\} \cup \{xy|x, y \in L\} \dots \tag{19}$$

Therefore we have an algorithm that goes through all the words now since the alphabet of P is enumerable, then P^{∞} and its subsets are enumerable.

Lemma 4 –Suppose that R is an enumerable set and f is a computable function which is defined on all elements of R. Then $f(R)$ is an enumerable set.

► *proof:* If R is empty, then so is $f(R)$ therefore enumerable by definition. If R is enumerated by a function $\rho(x)$ then $f(R)$ is enumerated by $\eta(x) = f(\rho(x))$.

Theorem 5 — The set of all provable words (for a deductive system) is enumerable.

► *proof:* Let Q be the set of all provable words for the deductive system $\langle P, \bar{P}, \delta \rangle$ obviously $Q = \delta(\bar{P})$. However P is enumerable by theorem 4. Hence Q is also enumerable by lemma 4.

It follows now that if T is not enumerable set, then it is impossible to find a complete and consistent deductive system for the pair $\langle \mathscr{L}, T \rangle$, since the set Q of provable words for any consistent deductive system is a proper subset of T, and there will have to be an element on complement $T \backslash Q$. Such an element is a true but unprovable statement!

Theorems 3 and 5 together give a condition on a fundamental pair which is necessary and sufficient for the existence of a complete and consistent deductive system for the pair. This condition is enumerability of the set of all truths.

One would expect that in a "rich" or "expressive" language the set of all truths is too complicated to be enumerable, and hence there are no complete and consistent deductive systems for such a language. However, the criterion in theorems 3 and 5 is not very convineint to use, since it is often difficult to study the entire set T.

4.2 Philosophy

In this section we would try to understand the incompleteness theorems out of the formal mathematical language. We would then proceed to talk about a mathematical universe.

4.3 Introduction

Gödel's incompleteness theorems are groundbreaking outcomes in present-day logic that transformed the comprehension of mathematics and logic, and had significant implications for the philosophy of mathematics. While some philosophers have attempted to utilize these outcomes in other areas of philosophy, the validity of many such applications is often debatable.

To comprehend Gödel's theorems, it is imperative to clarify crucial ideas related to them, including "formal system," "consistency," and "completeness." Essentially, a formal system signifies a collection of axioms furnished with inference rules, which enable the generation of new theorems. The axioms must be finite or at least decidable, meaning that an algorithm can determine whether a given statement is an axiom or not. If this criteria is fulfilled, the theory is known as "recursively axiomatizable," or simply "axiomatizable." The inference rules of a formal system are also effective operations that allow one to ascertain if one has made a legitimate application of a rule of inference. As a result, it is always possible to determine if any given finite sequence of formulas constitutes a real derivation in the system, given the axioms and the rules of inference.

A formal system is complete if every statement of the language of the system can be derived (proven) in the system or its negation can be proven. A formal system is consistent when there exists no statement in which the statement itself and its negation can both be derived in the system. In this context, only consistent systems are of interest since an inconsistent formal system renders every statement derivable, hence rendering such a system trivially complete.

Gödel discovered two distinct yet interconnected incompleteness theorems, typically known as the first and second incompleteness theorems. Although "Gödel's theorem" may refer to both in conjunction, it often refers to the first one separately.

With a modification introduced by J. Barkley Rosser in 1936, the first theorem can be roughly stated as follows:

> "Any consistent formal system F within which a certain amount of elementary arithmetic can be carried out is incomplete, meaning there are statements in the language of F which can neither be proved nor disproved in F."

Gödel's theorem not only asserts the existence of such statements, but also explicitly produces a specific sentence that is neither provable nor refutable in formal system F through Gödel's proof method. This "undecidable" statement can be mechanically generated from a specification of F and is a relatively simple statement of number theory, a purely universal arithmetical sentence.

One common misconception about Gödel's first theorem is interpreting it as showing the presence of truths that cannot be proven. However, this is incorrect because the incompleteness theorem deals with derivability in a particular formal system, rather than provability in an absolute sense. For any statement A unprovable in a given formal system F, there are other formal systems where A is provable (by taking A as an axiom). On the other hand, the standard axiom system of Zermelo-Fraenkel set theory (denoted as ZF or ZFC with the axiom of choice) is powerful enough to derive all ordinary mathematics. Nevertheless, there exist arithmetical truths that are not provable even in ZFC due to Gödel's first theorem. Hence, proving them would require a formal system that goes beyond ZFC methods. Therefore, these truths are not provable using today's "ordinary" mathematical methods and axioms, nor can they be proved in a way that mathematicians would regard as conclusive and unproblematic.

Gödel's second incompleteness theorem concerns the limitations of consistency proofs and can be roughly stated as:

> "For any consistent system F within which a certain amount of elementary arithmetic can be carried out, the consistency of F cannot be proved in F itself."

The second theorem requires a slightly more elaborate version of formal system F, encompassing more arithmetic than the one required for the first theorem, which holds under much weaker assumptions. It is important to note that, like the first incompleteness theorem, the second theorem pertains only to formal provability or derivability relative to some formal system (in this case, F itself). It does not imply anything about whether the statement "T is consistent" can be proven true by an argument considered conclusive or acceptable by mathematicians for a particular theory T satisfying the conditions of the theorem. For many theories, such proof is possible.

4.4 Incompleteness Theorems and The Church-Turing Thesis

Initially, Gödel demonstrated the incompleteness of a particular but extensive formalized theory P, which is a variation of Russell's type-theoretical system PM (Principia Mathematica). This system encompasses all extensions of P with the same language whose set of axioms is primitive recursive. Although Gödel did not demonstrate, he suggested that the proof could be adapted to apply to standard axiom systems of set theory such as ZFC. At that time, it was unclear how general the result was, despite the fact that Gödel had a very general outcome.

What remained uncertain was the analysis of the intuitive notion of decidability, necessary in characterizing the concept of an arbitrary formal system. Mathematicians and logicians have employed the intuitive notion of a decision method since ancient times. For the general results such as the general incompleteness theorems or the undecidability results, however, a precise mathematical explication of the notion would be necessary. Instead of decidable sets or properties, effective or computable functions or operations are often considered, but these are two sides of the same coin.

Gödel, Church, and Turing independently proposed different exact mathematical definitions of computable functions and decidable sets of numbers, which were later found to be equivalent. Turing's careful conceptual analysis that used fictional and abstract computing machines, conventionally known as "Turing machines," was significant, as emphasized by Gödel himself. The intuitive notion and some of these mathematical explications are often labeled as "The Church-Turing thesis." For historical reasons, the term "recursive function" has been dominant in the logical literature, so decidable sets are often termed as "recursive sets."

To comprehend the incompleteness and undecidability results correctly, understanding the distinction between two key concepts regarding sets is crucial. First, there may be a mechanical method that decides whether any given number belongs to the set or not, in which case, the set is referred to as "decidable" or "recursive." Second, there may be a mechanical method that generates or lists the elements of the set, number by number. In the latter case, the set is referred to as "recursively enumerable" (r.e.), meaning it can be effectively generated or is "semi-decidable." A fundamental outcome of the theory of computability is that semi-decidable sets can be effectively generated but not decidable. This is essentially the essence of the first incompleteness theorem at an abstract level. However, if both a set and its complement are recursively enumerable, the set is recursive, i.e., decidable.

5 Nature and the Incompleteness

The incompleteness theorem, together with the computation theorem (Church-Turing Thesis) establish a limit on mathematical abstraction. Being depended on the choice of axioms the two refer to unprovable or incomputable entities in mathematics and computations. There can be a lot to learn from these two theorems and to grasp a better understanding of the world as a whole. But there is a subtle difference between physicists and mathematicians.

From the basic philosophies to the moder science we always assumed deterministic behaviour for the nature, and so we should, since the classical physics is deterministic and also .

References

[1] Wikipedia contributors. Mathematical induction — Wikipedia, the free encyclopedia, 2023. [Online; accessed 5-June-2023].