Carolina Math Club Logic Talks - Spring 2025

Notes by Alex Paschal

Contents

1 Introduction

These are notes for two talks on logic given through the Carolina Math Club at the University of North Carolina at Chapel Hill in the spring of 2025. My hope is that these talks expose more students to the basics of logic, as there is currently only one logic course (PHIL 455) offered biyearly which covers these topics.

In two lectures, we will prove the semantic version of Gödel's first incompleteness theorem, which states that any sufficiently complex arithmetic system contains true statements which are unprovable. We will touch on other popular issues in logic such as computability and paradox. Here is a tentative weekly schedule:

Week 1: Discuss the philosophical problem of paradox by examining Russel's paradox, the liar paradox, and Richard's paradox. Define and discuss the basic properties of axiomatized logical systems and Peano arithmetic.

Week 2: Discuss computability, including a presentation of the halting problem. Present Gödel numbering in arbitrary arithmetic systems. Prove the first incompleteness theorem modulo the assumption that Peano arithmetic can express and capture Prf.

2 Paradoxes

The following definition appears on the back of [2].

Definition 2.1. *A* paradox *is an unacceptable conclusion derived by apparently acceptable reasoning from apparently acceptable premises.*

The word "unacceptable" is crucial here: it tells us that we should strive to resolve paradoxes when they arise.

2.1 Russell's Paradox

The following is math's most famous (set-theoretic) paradox, published by Bertrand Russell in 1901.

Paradox 2.2 (Russell's paradox)**.** *There are sets that are not members of themselves, like the set of all sets which have less than 10 elements. There are also sets that are members of themselves, like the set of all imaginable sets. Let R be the set of all set which are not members of themselves. Is $R \in R$?*

Let's analyze. Assuming that $R \in R$, we get, by the definition of R, that $R \notin R$, and similarly, if we assume that $R \notin R$ (and thus $R \in R^c$), the definition of R tells us that $R \in R$. Therefore, $R \in R$ iff $R \notin R$: a contradiction.

In ZFC (Zermelo-Fraenkel set theory with the axiom of choice included), Russell's paradox is evited by the axiom schema of separation, which states in one instance that given a formula $\varphi(x)$ and a set A, there exists a set $S \subseteq A$ which contains all the elements $x \in A$ for which $\varphi(x)$ holds. How does this resolve the paradox? Because, since the set of all sets violates this axiom, it (and thus R) cannot be a set in ZFC.

The obvious problem in Russell's paradox is self-reference: that a set can be a member of itself is where the issue starts. This motivated Alfred North Whitehead and Bertrand Russell to write *Principia Mathematica* wherein they attempted create a mathematical system which is free from self-reference. Although they were unsuccessful, their effort highlights the relative simplicity of formal (i.e., mathematical) logic in comparison to informal logic, wherein one cannot simply point to self-reference as an issue.

2.2 The Liar Paradox

The following paradox is so pernicious that it drove the ancient Greek scholar Philetas to insomnia and eventually death.

Paradox 2.3 (The liar paradox)**.** *This sentence is a lie.*

A quick analysis reveals the issue. Let φ be the sentence "φ is false." If we assume that φ is true, it tells us that it is false, and vice versa. Thus, φ is true iff it is false: a contradiction.

Unlike Russell's paradox, it's not so clear how to remedy the liar paradox. The man on the street would blame self-reference, but according to [2], that's both "incorrect" and "inadequate." Why? It's incorrect because there are non-paradoxical self-referential sentences, such as this one. And it's inadequate because one can construct a non-self-referential liar paradox, as seen below.

(said by α on Monday) Everything β will say on Tuesday is true.
(said by β on Tuesday) Nothing α said on Monday is true.

Further, is the fact that the liar paradox is neither true nor false *really* problematic? There exist non-paradoxical sentences which are neither true nor false, such as,

"You should stop beating your dog."

Point being, informal logic is complicated. I encourage you to look further into these issues, just don't end up like Philetas.

2.3 Richard's Paradox

Paradox 2.4 (Richard's paradox)**.** *Some phrases in natural language define properties of natural numbers (e.g. "divisible by exactly two natural numbers") and some do not (e.g. "is a city in France"). There is an infinite list of English phrases which define properties of natural numbers. Order them by length, and then order phrases of equal length lexicographically. This yields an infinite list of phrases: $r_1, r_2, \ldots$. For each $n \in \mathbb{N}$, we can ask if n satisfies the property r_n. We will say that n is Richardian iff it doesn't satisfy the property r_n. But then, to be Richardian is a property of natural numbers, and so appears somehwere in our list, say, as r_N. Now we ask, is N Richardian?*

The paradox here is quite similar to the two previous ones. Say N were Richardian: then it wouldn't satisfy r_N, so it wouldn't be Richardian. Alternatively, if N were not Richardian, it would satisfy r_N and so would be Richardian. Thus, N is Richardian iff it is not Richardian: a contradiction.

We resolve this paradox by noting that the property of being Richardian is not purely arithmetic (since it relies on the natural language we use to discuss arithmetic properties) and thus does not belong in the list we described.

By carefully distinguishing between mathematics and metamathematics, Gödel used a similar argument to Richard's paradox to prove his first incompleteness theorem, as we'll soon see.

3 Logical Systems

3.1 Basic Definitions

In a typical logic course this section would take several weeks, but given our time constraints we'll do without most of the formality. A *logical system* is a system of reasoning with statements that are either true or false. These statements come from the system's *formal language*, which consists of a *syntax* and *semantics*. The syntax determines which statements are *well-formed formulas*, or *wff*s. For example, $\forall x \forall y (x + y = y + x)$ is a wff but $\forall \implies (x + \times = S($ is not. The semantics of the language determines the meaning of wffs. Traditionally, it would say that $\forall x (x + 0 = 1)$ is false while $\forall x (x + 0 = x)$ is true. The system is also endowed with a *deductive system*, which includes

axioms and *rules of inference*. Axioms are statements that we take to be true without need for proof while rules of inference allow us to take two wffs, say P and Q, and combine them: for example, $P \implies P \vee Q$.

3.2 L_A and Peano Arithmetic

We denote by L_A the *language of arithmetic* ("language" in the sense of the above definition). L_A has the following non-logical symbols: 0, S, $+$, $\times$.[1] It also includes the traditional logical symbols like $\implies$ and $\neg$, as well as letters used to create variables. From it we can define Peano arithmetic (PA) by the following $6 + \infty$ axioms:

1. $\forall x(\mathrm{S}x \neq 0)$,
2. $\forall x \forall y(\mathrm{S}x = \mathrm{S}y \implies x = y)$,
3. $\forall x(x + 0 = x)$,
4. $\forall x \forall y(x + \mathrm{S}y = \mathrm{S}(x + y))$,
5. $\forall x(x \times 0 = 0)$,
6. $\forall x \forall y(x \times \mathrm{S}y = (x \times y) + x)$,

∞. $[\varphi(0) \wedge \forall x(\varphi(x) \implies \varphi(\mathrm{S}x))] \implies \forall x \varphi(x)$.

Why does the last axiom, known as the *induction schema*, represent infinite many? Because you can plug in *any* wff φ to the formula and get a new axiom.[2]

I encourage you to think about how, from these axioms, we can prove every familiar fact about the natural numbers. To give an example, say we wanted to prove the following *riveting* result.

Proposition 3.1. *For all y, $0 + y = y$.*[3]

Proof.

(i) $\forall x(x + 0) = x$,

(ii) $0 + 0 = 0$,

(iii) $\forall x \forall y(x + \mathrm{S}y = \mathrm{S}(x + y))$,

(iv) $\forall y(0 + \mathrm{S}y = \mathrm{S}(0 + y))$,

(v) $\forall y(0 + y = y)$. □

This proof highlights something important: proofs in PA are just lists of wffs $\varphi_0, \varphi_1, \ldots, \varphi_n$ with $\varphi_n = \varphi$ such that each φ_i is an axiom, an assumption, or follows from the preceding statements by a rule of inference.

[1] S denotes the *successor function*, which we use in writing down the natural numbers as follows: $1 = \mathrm{S}0$, $2 = \mathrm{SS}0$, etc. In general, we denote by $\overline{n}$ n occurences of S followed by a 0.

[2] It should be noted that φ has to "have x free." What this means is that you can plug in values for x: for example, the wff $x = 0$ has x free but $\forall x(x = x)$ does not.

[3] This is indeed a new result since we're not given that addition is commutative.

4 Outlining a Gödelian Proof

4.1 Gödel Numbering

Gödel invented a way to give syntactic objects numerical codes. First, we fix a series of *basic codes* for the symbols and variables of L_A:[4]

$\neg$	$\wedge$	$\vee$	$\Longrightarrow$	$\forall$	$\exists$	$=$	$($	$)$	0	S	$+$	$\times$	x	$'$
2	3	5	7	11	13	17	19	23	29	31	37	41	43	47

Now, suppose we have an expression φ which is a sequence of k symbols and/or variables $s_1, s_2, \ldots, s_k$. To calculate φ's Gödel number (g.n.), which we denote by $\ulcorner\varphi\urcorner$, we find the basic code c_i for each symbol/variable s_i, and then, denoting the i-th prime by p_i, let $\ulcorner\varphi\urcorner = 2^{c_1}3^{c_2}\cdots p_i^{c_i}$. For example:

(i) The single symbol S has g.n. 2^{31}.

(ii) PA's 3rd axiom has g.n. $2^{11}3^{43}5^{19}7^{43}11^{37}13^{29}17^{17}19^{43}23^{23}$.

By the fundamental theorem of arithmetic, this numbering scheme gives a one-to-one correspondence between expressions in L_A and a subset of the natural numbers.[5]

While coding individual expressions is useful, we'd really want to talk about whole proofs via their code numbers. For this, we introduce *super Gödel numbers*:

Definition 4.1. *Suppose $\varphi_1, \varphi_2, \ldots, \varphi_k$ is a sequence of expressions in PA with corresponding g.n.'s $g_1, g_2, \ldots, g_k$. Then, the* super Gödel number *which encodes this sequence is $2^{g_1}3^{g_2}\cdots p_k^{g_k}$.*

I won't give any examples of super g.n.'s because they quickly become unruly, but hopefully you see that a) there is, again, a one-to-one correspondence between super g.n.'s and sequences of expressions in L_A and b) the set of super g.n.'s is disjoint from the set of g.n.'s. Thus, given a g.n. or super g.n., we can find the unique expression or sequence of expressions in L_A to which it corresponds, and vice versa.

4.2 Arithmetizing Properties and Relations

Now that we can talk about whole proofs via their code numbers, the following relation becomes very useful:

Definition 4.2. $\mathrm{Prf}(m, n)$ *holds iff m is the super g.n. of a PA-proof of the sentence with g.n. n.*

The key result about this relation, which we will assume in this chapter and prove later, is the following:

Theorem 4.3. *PA can express and capture* Prf *by a wff.*

In short, this means that this relation is not metamathematical, as being Richardian was. So, whereas being Richardian created a paradox, Prf can be used to prove the first incompleteness theorem without violating the distinction between mathematics and metamathematics.

[4] Here we use x and $'$ to create infinite many variables: x, x', x'', etc.

[5] It is a proper subset because 100, say, has no corresponding expression in L_A (why?).

4.3 The Proof

Diagonalization is a recurring theme in our paradoxes, and Gödel, too, uses it.

Definition 4.4. *The diagonalization of a wff φ with one free variable is the wff* $\text{diag}(\varphi) = \varphi(\overline{\ulcorner \varphi \urcorner})$.

For example, the diagonalization of $\varphi(x) = \exists y F(x, y)$ is $\text{diag}(\varphi) = \exists y F(\overline{\ulcorner \exists y F(x, y) \urcorner}, y)$. Why is this substitution operation called "diagonalization?" Let's think back to Richard's paradox. Properties of natural numbers were listed out in a certain order and then we was asked if n satisfied the n-th property. In a similar fashion, Gödel numbering gives us a way to list out wffs, so we can ask if the wff with g.n. n holds for n.

Per usual, the diag operation gives rise to a relation:

Definition 4.5. *The relation* $\text{Diag}(m, n)$ *holds iff m is the g.n. of the diagonalization of the wff with g.n. n.*

Let's combine the previous two definitions:

Definition 4.6. *The relation* $\text{Prfd}(m, n)$ *holds iff m is the super g.n. for a PA proof of the diagonalization of the wff with g.n. n.*

Now we can cut to the chase. Define the open wff U ("U" for "unprovable") by $U(y) = \neg\exists x\, \text{Prfd}(x, y)$ and then diagonalize U to get $G = U(\overline{\ulcorner U \urcorner}) = \neg\exists x\, \text{Prfd}(x, \overline{\ulcorner U \urcorner})$. We get the following:

Theorem 4.7. *G is true iff it is unprovable in PA.*

Proof. Consider what it means for G to be true. G is true iff there is no number m such that $\text{Prfd}(m, \overline{\ulcorner U \urcorner})$; i.e., there is no number m such that m encodes a PA proof for the diagonalization of the wff with g.n. $\ulcorner U \urcorner$, which is G. □

This proves (the semantic version of) Gödel's first incompleteness theorem for Peano arithmetic, and a similar proof can be used for *any* sufficiently complex arithmetic system.[6]

5 Computability

5.1 Basic Notions

In general, the question of computability is the following: given a specific problem, does there exist an algorithm which can solve it in finite time? If so, said problem is called *effectively decidable*; if not, it's called *undecidable*.

To give some examples, the truth of a logical sentence containing $\implies$, $\wedge$, $\neg$, etc., but not $\forall$ and $\exists$, is effectively decidable via proof tables. If we're given $P \implies (Q \vee R)$, say, we can write out a table with all possible values of P, Q, R, $Q \vee R$, and $P \implies (Q \vee R)$ in finite time since there are only finitely many (in this case, 2^3), and thus determine if $P \implies (Q \vee R)$ is a tautology. On the other hand, checking if a given element is a member of an arbitrary infinite set is undecidable, since the best we can do is loop through the set's members and check if they're equal to the given element.

[6] Okay, this is a lie. This only holds for effectively axiomatized formal theories, as mentioned in Computability. Those who are curious should read through [4] for more information.

If none of them are, the algorithm will run forever, and if one of them is, we are only guaranteed to receive a response in finite time if the set is countable.

Now to clarify the setting we're working in. For an arbitrary logical system containing enough arithmetic to express $+$ and $\times$, we want our set of axioms to be effectively decidable, for $+$ and $\times$ to be effectively computable (meaning we can compute $4+3$, say, in finite time), and for our deductive system to be effectively formalized (meaning we can determine the validity of a logical statement in finite time). Such a system is called an *effectively axiomatized formal theory.*

5.2 The Halting Problem

One may ask if the problem of determining whether or not a program will stop is effectively decidable. This is known as the halting problem.

Theorem 5.1. *The halting problem is undecidable.*

My favorite proof of this theorem, which you can read here, is "Scooping the Loop Snooper," written in Dr. Seuss prose. I'll present a less flowery argument.

Proof. Assume, for sake of contradiction, that the halting problem is decidable via an algorithm P; i.e., given a program, P can predict whether or not the program will halt. Define Q as follows: given a program, if P says the program will halt, then Q runs forever, and vice versa. Now, plug Q into P. If P says Q will stop, then Q will run forever, and if P says Q will not stop, then Q will stop: a contradiction. □

6 Primitive Recursive Functions

Here I will define primitive recursive (p.r.) functions and outline their usage in proving Theorem 4.3. For a complete exposition, I recommend reading chapters 10 and 11 in [4], since writing out in sufficient detail how the proof uses them would amount to copying them down verbatim.

6.1 Defining Primitive Recursive Functions

The class of p.r. functions is built from the following *initial functions.*

Definition 6.1. *The* initial functions *are the successor function* S*, the zero function Z, and the place functions $I_i^k(x_1, x_2, \ldots, x_k) = x_i$ for all $k \geq 0$, $1 \leq i \leq k$.*

From these functions, we can generate new functions in two ways. First, we have standard composition of functions. If $\vec{x}$, $\vec{y}$, and $\vec{z}$ represent tuples of variables and $g(\vec{y})$ and $h(\vec{x}, u, \vec{z})$ are functions, then the function $f(\vec{x}, \vec{y}, \vec{z}) = h(\vec{x}, g(\vec{y}), \vec{z})$ is defined *by composition* by substituting g into h. We can also define functions *by primitive recursion* as follows:

Definition 6.2. *Suppose the following holds:*

$$f(\vec{x}, 0) = g(\vec{x}),$$
$$f(\vec{x}, \mathrm{S}y) = h(\vec{x}, y, f(\vec{x}, y)).$$

Then f is defined from g and h by primitive recursion*.*

As an example, consider the factorial function. In this case, $\vec{x}$ is empty and we have

$$0! = 1 \text{ and } (\mathrm{S}y)! = y! \times \mathrm{S}y,$$

so the factorial function is defined from the "one function" (which maps $x \mapsto 1$ for all x) and the $\times$ function by primitive recursion. Similarly, as one does in PA, the $\times$ function is defined by primitive recursion from the zero function and the + function, which is defined by primitive recursion from the zero function and the successor function S. Generalizing a little bit, we see that for any p.r. function f, there exists a *definition chain* $f_0, f_1, \ldots, f_n$ where each f_i is either an initial function or is defined from previous functions in the sequence by composition or primitive recursion and $f_n = f$.

6.2 Outlining the Proof that Peano Arithmetic Can Express and Capture Prf

Theorem 4.3 states that PA can "express" and "capture" Prf. PA *expresses* Prf by the wff $\varphi(x, y)$ if, for any m and n, $\mathrm{Prf}(m, n)$ iff $\varphi(\overline{m}, \overline{n})$. PA *captures* Prf by the wff $\varphi(x, y)$ if, for any m and n, $\mathrm{Prf}(m, n)$ implies that PA can prove $\varphi(\overline{m}, \overline{n})$ and $\neg\,\mathrm{Prf}(m, n)$ implies that PA can prove $\neg\varphi(\overline{m}, \overline{n})$.

How do we prove Theorem 4.3? We first prove that PA can express and capture all p.r. functions. Since each p.r. function has a definition chain, it suffices to prove the following:[7]

1. PA can express and capture the initial functions.
2. If PA can express and capture the functions g and h, then it can also express the composition of g and h.
3. If PA can express and capture the functions g and h, then it can also express the function f which is defined by primitive recursion from g and h.

We then prove that Prf is a p.r. relation in the following sense, thus completing our proof.

Definition 6.3. *The* characteristic function *of a two-place arithmetic relation R is defined by*

$$c_R(m, n) = \begin{cases} 1 & m \textit{ is related to } n \\ 0 & m \textit{ isn't related to } n. \end{cases}$$

We say that R is a p.r. relation *if c_R is a p.r. function.*

How is this done? One can prove the following:

Theorem 6.4. *If a function can be computed by a program without open-ended searches (e.g., with basic bounded "for" loops) then it is p.r.*

Thus, it suffices to describe a program which computes Prf and contains no open-ended searches. Let's think about what $\mathrm{Prf}(m, n)$ does. First, it finds the PA-proof $\varphi_0, \varphi_1, \ldots, \varphi_n$ that m is the super g.n. of and the wff φ that n is the g.n. of. This contains no open-ended searches since, as PA is an effectively axiomatized formal theory, the operation $\times$ (and thus prime factorization) is effectively computable. Then, it checks if $\varphi_0, \varphi_1, \ldots, \varphi_n$ is a PA-proof of φ, which is effectively decidable since PA is an effectively axiomatized formal theory, so our program contains no open-ended searches.

[7] (1) and (2) are relatively easy to prove while (3) takes lots of work in both cases.

7 Conclusion

I hope these talks and/or notes have made you excited to learn more about logic. If paradoxes interested you, I highly recommend [2] as well as this article on paradoxes in the Stanford Encyclopedia of Philosophy. For more on Gödel's incompleteness theorems, consider [1] for a slightly slower introduction than these notes provided and [4] to clear up details I didn't cover. If you're still curious after reading through [4], you could consider [3] for a more thorough coverage of introductory logic as a whole or begin to look into model theory (using, e.g., Marker's book). If the philosophy department keeps with their usual schedule, PHIL 455 (Symbolic Logic), the class that introduced me to logic, will be taught next spring.

8 Acknowledgments

I'd like to thank Dr. Jim Pryor for introducing me to logic in PHIL 455 as well as his help locating resources that I used to plan these talks. I'd also like to thank Dr. Mark Williams for his advice which guided how I structured these talks and notes, and for sharing this notes template with me. Finally, Paul Hamrick caught several errors which were present in these notes. All remaining errors are, of course, my own.

References

[1] E. Nagel and J. R. Newman. *Gödel's Proof*. NYU Press, 2001. URL `http://www.jstor.org/stable/j.ctt9qg4j9`.

[2] R. M. Sainsbury. *Paradoxes*. Cambridge University Press, 3rd edition, 2009. URL `https://www.cambridge.org/core/books/paradoxes/1634F4E927B9478D1538AB06E8861F4F`.

[3] P. Smith. *An Introduction to Gödel's Theorems*. Logic Matters, 2nd edition, 2013. URL `https://www.logicmatters.net/resources/pdfs/godelbook/GodelBookLM.pdf`.

[4] P. Smith. *Gödel Without (Too Many) Tears*. Logic Matters, 2nd edition, 2022. URL `https://www.logicmatters.net/resources/pdfs/GWT2edn.pdf`.