FORMALLY VERIFYING PEANO ARITHMETIC

by

Morgan Sinclaire

A thesis

submitted in partial fulfillment

of the requirements for the degree of

Master of Science in Mathematics

Boise State University

May 2019

BOISE STATE UNIVERSITY GRADUATE COLLEGE

DEFENSE COMMITTEE AND FINAL READING APPROVALS

of the thesis submitted by

Morgan Sinclaire

Thesis Title: Formally Verifying Peano Arithmetic

Date of Final Oral Examination: 15 March 2019

The following individuals read and discussed the thesis submitted by student Morgan Sinclaire, and they evaluated his presentation and response to questions during the final oral examination. They found that the student passed the final oral examination.

M. Randall Holmes, Ph.D.	Chair, Supervisory Committee
John Clemens, Ph.D.	Member, Supervisory Committee
Samuel Coskey, Ph.D.	Member, Supervisory Committee
Elena Sherman, Ph.D.	Member, Supervisory Committee

The final reading approval of the thesis was granted by M. Randall Holmes, Ph.D., Chair of the Supervisory Committee. The thesis was approved by the Graduate College.

For Aiur

ACKNOWLEDGMENTS

The main people who deserve counterfactual credit for this work include:

1. My parents. In particular, my mom and my dad.

2. My advisor. He *far* exceeded his administrative obligations with the number of hours he was available to me for help, both on this thesis and for all of my miscellaneous logic questions.

3. The other people here who have made Boise State a good place to learn logic: Professors Elena Sherman, John Clemens, and Sam Coskey, and the only other logic grad student, Gianni Krakoff.

ABSTRACT

This work is concerned with implementing Gentzen's consistency proof in the Coq theorem prover.

In Chapter 1, we summarize the basic philosophical, historical, and mathematical background behind this theorem. This includes the philosophical motivation for attempting to prove the consistency of Peano arithmetic, which traces itself from the first attempted axiomatizations of mathematics to the maturation of Hilbert's program. We introduce many of the basic concepts in mathematical logic along the way: first-order logic (FOL), Peano arithmetic (PA), primitive recursive arithmetic (PRA), Gödel's 2nd incompleteness theorem, and the ordinals below ϵ_0.

In Chapter 2, we give a detailed exposition of one version of Gentzen's proof. Gentzen himself gave many similar proofs of the consistency of PA, as did several others after him; we describe the version given in Mendelson [20]. In comparison to the latter, our formulation fills in many erstwhile omitted details that we feel the reader deserves to see spelled out. We also have made other minor rearrangements, but altogether have found little to improve on that classic work of exposition.

Chapter 3 is a detailed walkthrough of our present 5000-line implementation, with each section corresponding to the 11 sections of our code. There were three main conceptual challenges to implementing the chapter 2 proof: properly defining the ordinals below ϵ_0 and proving their basic properties, defining PA_ω's proof trees in a streamlined way, and defining the proof tree transformation operations discussed in section 2.4. We have successfully addressed these problems, as discussed in sections

3.2, 3.8, and 3.9 respectively. Our implementation is still incomplete as of this writing, but we substantiate our claim that the remaining work is largely routine.

In our concluding chapter, we consider the likely future directions of this work, and discuss its place in the current literature.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

Mathematics occupies a special place in our intellectual landscape as the only place where results can be believed with *certainty*; even physicists recognize their field as fallible, pointing to how the theory of relativity overturned centuries of evidence for Newtonian mechanics. Yet one single instance of the smallest proof can settle the biggest question–once, and forevermore. To every other discipline, mathematics is looked up to as the paragon of certitude.

Outsiders are often surprised to learn that about a century ago, almost nothing felt certain in mathematics, as rival schools of thought in the mathematical world battled to define the very core of their subject. This "foundational crisis of mathematics", as it became known, eventually subsided, but not until spawning an entire new branch of math known as mathematical logic. This area of study, with its subdivisions of set theory, model theory, computability theory, and proof theory, came to take on a life of its own in the mathematical community. However, this last subfield of proof theory was motivated quite directly to reason about the limits of what is provable, and to this day attempts to answer some of the hard questions about exactly what can be put under the scope of mathematical certainty.

Although the foundational contributions here are minor, this work nevertheless descends from these motivations, and cannot be understood without them. Below,

we sketch out this intellectual genealogy, introducing proof theory along the way.[1]

1.1 The Axiomatic Method

Whenever any human makes any kind of argument for some *conclusion*, that argument must rest on some *premise*[2] that is merely assumed, and the conclusion is only credible insofar as the premise(s) are. If the premise is called into question, there are only 3 possibilities:

1. The premise is justified by invoking the conclusion. This is called a *circular* argument.

2. The premise is justified by invoking a new premise, which is in turn justified by a new premise, and so on forever. This is called an *infinite regress*.

3. The premise is justified by invoking a new premise, which is in turn justified by a new premise, and so on until we reach some premise A that cannot itself be justified, but which feels self-evident (at least to some). A is called an *axiom*.

This basic philosophical observation, that every argument rests on either a circular argument, an infinite regress, or some axiom A, is known as Munchausen's trilemma and has been known for thousands of years.[3] Since the first two options are rarely advocated, almost all of the major thinkers in the Western intellectual tradition

[1] This story has been told many times at varying levels of accuracy. The fact is, these issues are complicated, subtle, and often misunderstood even by experts, and in the space of this chapter, we cannot hope to do full justice to many of the details. While we note our more egregious oversimplifications and omissions in various footnotes, even these can only go so far; readers desiring a more advanced and comprehensive treatment of these topics are recommended to consult [13], [30].

[2] Or multiple premises, but that is irrelevant here.

[3] Even though the term itself was not coined until 1968, after an 18th century satire of an impossibly heroic Baron Munchausen, who pulled himself out of a mire by pulling on his own hair.

have supported (3) in some form. Mathematical arguments (i.e. proofs) are not at all immune to Munchausen's trilemma, and so we cannot even be certain about mathematical conclusions (i.e. theorems) unless they are proved from axioms we are certain about.

In this regard, Euclid's axiomatic development of geometry stands out as by far the most significant achievement of its time. However, Euclid's precision in stating his axioms and proving his theorems is quite lax by modern standards, and his axioms did not cover any areas of math outside geometry. In the intervening two millenia, little progress was made to axiomatize even the fundamental properties of numbers (i.e. basic arithmetic), and the more ambitious project of putting all of mathematics on a firm foundation of axioms would have to await the more precise language of mathematical logic.

1.2 Mathematical Logic

Formal logic is based on the simple idea that any argument can be regarded as *valid* or *invalid*, depending on whether the conclusion truly follows from the premise(s), and that valid inferences tend have an identifiable structure that distinguishes them from invalid inferences. For instance, if for some propositions P, Q, we know that P implies Q (which we write $P \to Q$) and that P is true, then we also know Q is true. This is called a *rule of inference*, since it lets us take two premises ($P \to Q$ and P) and validly derive a conclusion (Q) no matter what P, Q actually mean as sentences. That is, the rule, which is known as *modus ponens*, can be identified simply by its structure:

$$\frac{P \to Q \qquad P}{Q} \text{ modus ponens}$$

Another rule of inference is called modus tollens, which says that if we know $P \to Q$ and that Q is false (which we write $\neg Q$, i.e. "not Q"), then we can validly infer P is false:

$$\frac{P \to Q \qquad \neg Q}{\neg P} \text{ modus tollens}$$

On the other hand, if we know $P \to Q$ and $\neg P$, then we cannot validly infer anything about Q. A common mistake for novices is to infer $\neg Q$ from those two premises, and this is an invalid inference (as careful thought will reveal).

In the mid 19th century, these observations began to finally become published by multiple philosophers and mathematicians. Most notably, George Boole's 1854 *Laws of Thought* [3] introduced an entire syntax to represent *propositional logic*, a formal system for distinguishing valid and invalid inferences. In modern terms, the language of propositional logic looks like:

Notation	**Meaning**
$P \to Q$	P implies Q
$P \leftrightarrow Q$	P if and only if Q
$\neg P$	not P
$P \wedge Q$	P and Q
$P \vee Q$	P or Q

Propositional logic, also known as *Boolean algebra* or *Boolean logic*,[4] went on to become physically realized in Boolean circuits, and ultimately would provide the theoretical foundation for computer science. Still, the 5 *logical connectives* in the

[4]Strictly speaking, these terms are not fully synonymous, but they were treated as such in the 19th century since the distinctions are subtle.

above table could only do so much, only capturing some of the more basic logical inferences. To formalize *all* mathematics, more powerful tools were needed.

1.3 First-Order Logic

Gottlob Frege (1848-1925) was the first to articulate what we now call *logicism*, the view that all mathematics can be grounded in pure logic [45]. While mathematics has always been recognized as having a very logical quality, the idea of *all* mathematics being reduced to formal logic was new and striking: mathematicians talk about all sorts of domain-specific subject matter like ellipses, quadratic functions, integrals, etc., which have actual content to them, so it would certainly seem that mathematicians need at least some properly *mathematical* knowledge and intuitions to do their job, and that their thoughts cannot be reduced to the workings of *and*'s and *if-then*'s.

But Frege was also among the first to discover that formal logic could be made more powerful than just Boole's propositional logic. In addition to the connectives listed above, he introduced the *quantifiers*:

Notation	**Meaning**
$\forall x P(x)$	for all x, property P holds of x
$\exists x P(x)$	for at least one x, property P holds of x

For instance, the classic (valid) inference:

1. (Premise 1): Socrates is a man

2. (Premise 2): All men are mortal

3. (Conclusion): Socrates is mortal

Cannot be captured in propositional logic. However, if we define the properties $P(x)$:= "x is a man" and Q := "x is mortal", then this inference can be formally represented in modern logic as follows:

$$\frac{\text{P(Socrates)} \qquad \forall x(\text{P(x)} \rightarrow \text{Q(x)})}{\text{Q(Socrates)}}$$

In this way, propositional logic matured under Frege's genius into *first-order logic* (FOL),[5] which is more powerful but also more sophisticated. In modern terms, to define FOL, we first define the notion of a *term*:

1. Variables $x, y, z, ...$ are terms (we will often denote these $x_0, x_1, x_2, ...$ to ensure we do not run out of variable names).

2. Constants $c_0, c_1, c_2, ...$ are terms (e.g. in the above example, "Socrates" is a constant)

3. If we have a function f and $t_1, t_2, ..., t_n$ are terms, then $f(t_1, t_2, ..., t_n)$ is a term.

For instance, we will regard addition + as a function when we formalize arithmetic in FOL. Then x_1 and x_4 are terms, so $x_1 + x_4$ will also be a term. An example of a constant will be 0, and we could also regard other numbers such as 7 or 53 as constants.[6]

An *atomic formula* is anything of the form $t_1 = t_2$, where t_1, t_2 are terms. For instance, $5 = 5$ and $3 = 2$ are both atomic formulas, which can either be true or false.

[5] Actually, FOL is a specific kind of logic with quantifiers, where the "first-order" indicates that we quantify only over variables, and not propositions themselves, i.e. we can say $\forall x$ but not $\forall P$. The technical advantages of using first-order logic were not known to Frege and were only widely recognized around 1930. For simplicity we do not describe this here, but the interested reader can consult `plato.stanford.edu/entries/logic-firstorder-emergence/`

[6] As we will see later, we will actually only need 0 as a constant symbol.

$2 + 2 = 4$ is also an atomic formulas, since $2 + 2$ and 4 are both terms. If we include multiplication as a function, we can also have atomic formulas like $4 + 8 = 3 \cdot 4$ or $4 \cdot 4 \cdot 4 = 15 \cdot 2 + 7 \cdot 5$.

In addition, $3 + x_0 = 5$ or $5 \cdot x_0 = x_1 + 4$ are atomic formulas, but unlike our previous examples, these do *not* have definite truth values. Whereas we can determine whether $4+8 = 3 \cdot 4$ is true or false, we cannot say the same about $3+x_0 = 5$ without further context: we have to know what x_0 actually is. We call x_0 here a *free variable* because this context is missing. This is important from a logical point of view because if an atomic formula has a free variable, then it does not express a specific proposition that we can regard as either true or false. We will say an atomic formula is *closed* if it has no free variables. We will say the same about terms, e.g. $9 \cdot 2$ is a closed term since it has no free variables, while $x_1 + 8$ is *not* a closed term since it has x_1.

With that mind, a *formula* is what you get when you start with atomic formulas, and apply logical connectives/quantifiers to them. More precisely, a formula in FOL is either:[7]

1. An atomic formula

2. $\neg A$, where A is a formula

3. $A \to B$, where A, B are both formulas

4. $\forall x_i A(x_i)$, where A is a formula and x_i is any variable.

[7]If we want, we can also build formulas out of the other connectives $\vee, \wedge$, and $\exists$. However, it turns out that this is unnecessary, since we can represent those using just the connectives $\neg, \to$, and $\forall$. This is because for any A, the formula $\exists x_0 A(x_0)$ is logically equivalent to $\neg\forall\neg A(x_0)$, and so we can regard the symbol $\exists$ merely as an abbreviation for $\neg\forall\neg$. Similarly, $A \vee B$ as well as $A \wedge B$ can both be written as a (more complicated) formula involving just $\neg$ and $\to$, although this is a technical point which we will not prove here.

So for instance, $5 = x_0$ is a formula by (1) since its an atomic formula, which means $\neg(5 = x_0)$, which we will write as $5 \neq x_0$, is also a formula by (2). Applying (3) to this, $5 \neq x_0 \to 2 \cdot 3 = 6$ is another formula. Finally, $\forall x_0(5 \neq x_0 \to 2 \cdot 3 = 6)$ is also a formula by (4).

As with terms and atomic formulas, we will say a formula is *closed* if it has no free variables, and consequently, it is only the closed formulas that have a definite truth value. For instance, neither $5 = x_0$ nor $5 \neq x_0 \to 2 \cdot 3 = 6$ are closed. On the other hand, in

$$\forall x_0(5 \neq x_0 \to 2 \cdot 3 = 6)$$

we will say the variable x_0 is *bound* by the universal quantifier $\forall x_0$, and is *not* free. Now that x_0 is bound, this formula now has a definite truth value (namely, it is true, because $2 \cdot 3 = 6$ is always true for every possible value of x_0).

In modern terms, the axioms of FOL, which we will use for the remainder of this work, are as follows:

$(FOL1)$ $\quad A \to (B \to A)$

$(FOL2)$ $\quad (A \to (B \to C)) \to ((A \to B) \to (B \to C))$

$(FOL3)$ $\quad (\neg B \to \neg A) \to ((\neg B \to A) \to B)$

$(FOL4)$ $\quad (\forall x A(x)) \to A(t)$ (if t is a closed term)

$(FOL5)$ $\quad (\forall x(A \to B)) \to (A \to \forall x B)$ (if x is not a free variable in A)

In addition, FOL has two rules of inference. The first is *modus ponens*, which we

saw earlier:

$$\frac{P \to Q \qquad P}{Q} \text{ modus ponens}$$

The other is *universal generalization*. This says that if we have proved that some property P holds for some arbitrary x (i.e. where x is a free variable) then we have proved P holds for all x:

$$\frac{P(x)}{\forall x P(x)} \text{ universal generalization}$$

(1-3) are axioms expressible in pure propositional logic, and FOL extends this with (4-5). These latter two add significant expressive power to mathematical logic. This made FOL a system more worthy of carrying out Frege's logicist program. But to do so, he needed one more important ingredient: set theory.

1.4 Set Theory

In mathematics, a set is simply a collection of objects. This, of course, is a very familiar notion outside of mathematics, but what Frege needed was a well-developed theory of *infinite* sets, and this is the nontrivial notion which the discipline of *set theory* refers to, and that did not exist at all before the 19th century.

There was good reason for this: infinite sets, to put it mildly, are strange, and in many cases do not behave at all like the finite sets we are accustomed to. For instance, Galileo (and others) [38] had noticed that even though the set of integers $\mathbb{Z} = \{..., -2, -1, 0, 1, 2, ...\}$ is "about twice as big" as the set of natural numbers $\mathbb{N} = \{0, 1, 2, ...\}$, in another sense these sets have the *same size*, since it is possible to put them in 1:1 correspondence:

$\mathbb{N}$	0	1	2	3	4	5	6	7	8	9	10	...
$\mathbb{Z}$	0	1	-1	2	-2	3	-3	4	-4	5	-5	...

Because of counterintuitive phenomena like this, Western intellectual thought, going back to the ancient Greeks, generally rejected infinite sets. Most followed Aristotle, who drew the distinction between *potential* and *actual* infinity, accepting the former while rejecting the latter.

To believe in potential infinity means to simply accept that for every number n, we can form a bigger number $S(n)$ (i.e. here $S(n)$ denotes the *successor* of n, which means $n+1$). This implies that there is no biggest number, that the natural numbers will never "run out", and instead are unending. We can say they are *never finished*, which in Latin corresponds to in- not + finitus 'finished', where we get the term *infinite*.

The idea of actual infinity arises from asking what happens if it *were* finished. Rather than treating the numbers as some specific individual (mathematical) objects that we can always obtain more of, believing in actual infinity is to collect *all* of these into a single set $\mathbb{N}$, and think about the mathematical properties this set might have:

$$\text{Potential infinity:} \quad 0, 1, 2, 3...$$

$$\text{Actual infinity:} \quad \{0, 1, 2, 3, ...\}$$

And strange things happen when one moves from potential to actual infinity, with many paradoxical results that were known even to the Greeks. Galileo himself apparently did not know what to make of his personal discovery that $\mathbb{N}$ seemingly both is and is not the same size as $\mathbb{Z}$.

However in the 1870s, the mathematicians Georg Cantor (1845-1918) and Richard Dedekind (1831-1916), in disregard of Aristotle's warnings and the received wisdom of two millennia of mathematical thought,[8] began to treat these infinite sets as *actual* objects with specific properties, founding the modern field of set theory. In particular, they would say that two sets have the same size, or *cardinality*, exactly when they can be put in 1:1 correspondence. For instance, $\mathbb{N}$ and $\mathbb{Z}$ have the same size under their definition, and they realized that even though it may be *counterintuitive* that they are equally big, this was not *contradictory*.

We say that some formal system is contradictory or *inconsistent* if, for some formula A, the system proves A and it also proves $\neg A$. Virtually all mathematicians and philosophers regard inconsistent systems as completely useless: we do not want our system to be able to prove a statement if it is not true,[9] and its not possible for A and $\neg A$ to both be true. Worse, since from two contradictory formulas it is possible to prove anything, such a system would also be uninteresting.

And so Cantor and Dedekind continued to work in their new set theory, they went further than Galileo or anyone had, proving more theorems that seemed strange, but never proving a contradiction. But there was one particular theorem Cantor proved that stood above the others: while one might expect that *all* infinite sets have the same size under their definition. While their cardinality concept might be internally consistent, it would not be that interesting if it said *every* infinite set is the same size, yet Cantor showed this was not the case.

---

[8]Strictly speaking, they were not the first, as Bolzano had made the case for actual infinities in work published posthumously in 1851. Still, this had little influence, and at any rate this stopped short of recognizing the cardinality concept or its importance.[8]

[9]Different philosophers might disagree about the exact meaning of "true", but in this context it is virtually unanimous that whatever "true" can reasonably mean, at least one of $\{A, \neg A\}$ is not true.

Dedekind and Cantor both considered $\mathbb{R}$, the set of all *real* numbers, which includes $0, 11, 4.3, \frac{7}{3}, \sqrt{2}, \pi, e, \sqrt{\pi + \frac{3}{e}}$, and can be thought of as the set of all numbers that can be represented by a possibly infinite decimal such as 37.835829437.... This is a more abstract set than $\mathbb{N}$ and is less easily visualized, since many of its members have an infinite decimal expansion with no particular pattern, but $\mathbb{R}$ is perhaps the only rival to $\mathbb{N}$ in terms of how often it is studied by mathematicians. And as the first demonstration of the power of set theory when they gave the first mathematically rigorous definitions of the real numbers. Their definitions were different but equivalent, and both described the reals in terms of the more familiar natural numbers. The catch was, their definition involved the *actual* infinity $\mathbb{N}$, and with only potential infinity, the cherished real numbers had no rigorous foundation.

Then Cantor went even further, and showed that $\mathbb{R}$ *cannot* be put in 1:1: correspondence with $\mathbb{N}$. In modern notation, while $|\mathbb{N}| = |\mathbb{Z}|$ since they can be put into 1:1 pairing, $|\mathbb{N}| < |\mathbb{R}|$, because Cantor proved that no matter how cleverly one rearranges the elements of $\mathbb{N}$ and $\mathbb{R}$, there will always be extra numbers left over in $\mathbb{R}$ that are not matched. This meant that their notion of size was not trivial, that not all infinities are the same, and that $\mathbb{R}$ is fundamentally bigger than $\mathbb{N}$.

Today, Cantor's result is considered among the great theorems of mathematics, but at a time when actual infinities were widely viewed as "too big" be even be coherent objects of study, Cantor faced open ridicule by many of his contemporaries, who often mistook their personal distaste of the new intellectual edifice with its logical merits, and called Cantor and Dedekind's work inconsistent. But his critics also included the leading mathematicians of the time, such as Leopold Kronecker (1823-1891) and Henri Poincare (1854-1912). While acknowledging the apparent consistency of these completed infinities (in spite of themselves), they dismissed the new set theory as

hollow and meaningless.

In any case, as mainstream mathematics became more abstract and general in the 19th century with certain developments in real analysis, abstract algebra, and topology, there grew a latent desire to talk about infinite sets like $\mathbb{N}$ and $\mathbb{R}$. The set theory of Cantor and Dedekind was ready-made to provide a rigorous foundation for mathematics well beyond just geometry or arithmetic, and over time mathematicians would notice this. And here, no one was more ahead of the curve than Gottlob Frege.

1.5 Basic Laws

Technically, Frege did not use Cantor and Dedekind's set theory *per se* but was certainly inspired by it, and actually developed a more complicated logical structure that similarly promised to fulfill his logicist dreams, and give a full axiomatization of all mathematics. For simplicity, we describe his *magnum opus* in modern set-theoretic terms. This was his *Grundgesetze der Arithmetik* ("Basic Laws of Arithmetic") [10], a landmark work in which, starting from just a few logical axioms, (his "Basic Laws") he carefully derived many of the fundamental laws of arithmetic. This 2-volume tome, published in 1893 and again in 1903, went all the way back to his 1879 book where he introduced the machinery we saw in 1.3, as well as his 1884 *Die Grundlagen der Arithmetik* ("The Foundations of Arithmetic") [9], which was a philosophical analysis of the concept of number.

But now, he had actually formalized arithmetic, now that he had extended his earlier work with his famous Basic Law V [10]:

[10]The actual statement of Basic Law V in Frege's work is actually more technical, resembling

$$\{x \mid P(x)\} = \{x \mid Q(x)\} \iff \forall x(P(x) \leftrightarrow Q(x))$$

But the much simpler statement given here is the actual import of the axiom in the context of

Axiom (Basic Law V)**.** *If P is some property, we can form the set of objects having that property:*

$$\{x \mid P(x)\}$$

For instance, if the property is "green", then we can form the set of all green things. Similarly, we can form the set of all even numbers, or the set of all sets with more than 5 elements.

At the same time, a young philosopher named Bertrand Russell (1872-1970) was coming around to similar ideas. As he set out on his own quest to reduce mathematics to pure logic, he began to play around with certain properties of sets. Noticing that if we regard any collection of objects as being a set, then some sets will be *members of themselves*, while others will not. For instance, then if we let U be the set of *all* sets, then U would itself be a set, and so by its definition U would be a member of U, i.e. $U \in U$.

Thus we can consider either a) the sets S which are members of themselves (i.e. $S \in S$) or b) the sets S which are not (i.e. $S \notin S$). Russell considered the latter collection of sets:

$$R := \{S \mid S \notin S\}$$

and asked, *Is R a member of itself?*

If $R \notin R$, then R is a set which is *not* a member of itself. But then R is a set that should be in R, so $R \in R$.

---

the rest of his formalism [45].

If $R \in R$, then R is a set which *is* a member of itself. But then R is a set that should *not* be in R, so $R \notin R$.

In other words, if $R \notin R$, then $R \in R$; but if $R \in R$, then $R \notin R$. Either way, we have reached a contradiction.

This result became known as *Russell's paradox*, and became noteworthy for how elegant it is, using only the concept of self-reference as well as $\notin$. But in particular, since membership $\in$ is a logical property, so is $\notin$. Thus, in Frege's Basic Law V, the logical property $P(x)$ can be taken to be $x \notin x$, and hence, in Frege's system, we can build Russell's set R, and derive a contradiction. In 1903, just as the 2nd edition of Frege's *Basic Laws* was going to press, he received a letter informing him that the entire framework he had been working on for 2 decades was inconsistent. As Russell reflected 60 years later [39, p. 127]:

> As I think about acts of integrity and grace, I realise that there is nothing in my knowledge to compare with Frege's dedication to truth. His entire life's work was on the verge of completion, much of his work had been ignored to the benefit of men infinitely less capable, his second volume was about to be published, and upon finding that his fundamental assumption was in error, he responded with intellectual pleasure clearly submerging any feelings of personal disappointment. It was almost superhuman and a telling indication of that of which men are capable if their dedication is to creative work and knowledge instead of cruder efforts to dominate and be known.

Frege promptly wrote an Appendix to his work, describing the derivation of contradiction within his own system. He then tried to restrict his Basic Law V

to be consistent, while still being powerful enough to derive arithmetic as he wanted, but in vain.

Russell's paradox itself, in the way it uses unboundedly large sets, came to be seen as the Achilles heel of set theory, even of the axiomatic method itself. In the aftermath, other antinomies of infinite sets became more widely known, and critics were emboldened. Set theory was not even internally consistent, as Cantor and Dedekind had believed; here, finally, was the actual contradiction. As Poincare wrote in "The Last Efforts of the Logicists" (emphasis in original) [25, pp. 193-195]:

> Logic therefore remains barren, unless it is fertilized by intuition...Logicism is no longer barren, it engenders antinomies. It is the belief in the existence of actual infinity that has given birth to these non-predicative definitions... *There is no actual infinity.* The Cantorians forgot this, and so have fallen into contradiction.

1.6 Last Efforts

Russell, for his part, continued where Frege gave up, and together with A.N. Whitehead, went on to publish *Principia Mathematica* (*PM*) [41], an even more ambitious attempt than Frege's. *PM* developed a new approach now known as *type theory*, which, roughly speaking, puts objects into certain collections called *types*, where we write x : `T` to indicate "x is of type `T`". For instance, we might write 5 : `nat` to indicate that 5 is a natural number, $\{3, 16\}$: `set(nat)` to mean $\{3, 16\}$ is a set of natural numbers, and $\{\{0, 8\}, \{1, 6, 9\}\}$: `set(set(nat))` to mean $\{\{0, 8\}, \{1, 6, 9\}\}$ is a set of sets of natural numbers. For notational convenience, we can say that these objects are, respectively, *type 0, type 1*, and *type 2*.

Types are different from sets in that every object has exactly one type. In the above example, while we have $\{3, 16\}$: `set(nat)`, we will never have $\{3, 16\}$: `nat`, nor $\{3, 16\}$: `set(set(nat))`. One consequence is that this system disallows objects like $\{2, \{6, 11\}\}$: this is not of type `set(nat)`, nor of `set(set(nat))`. We say this object is not *well-typed*, or that it fails to *type-check*, and so we cannot build it in our system. In general, a type $n + 1$ object is a set whose members are of type n.

In this way, roughly speaking, type theory prevents the construction of even the statement $S \notin S$ which is essential to Russell's paradox, because we can only say $X \in Y$ when X is of type n and Y is of type $n+1$, which is impossible when $X = Y$. In this way, Russell and Whitehead showed that this scourge of Frege's system was not a problem in theirs.[11]

Nevertheless, they were unable to *prove* that their system was consistent, and even though no one could find any contradictions, this did not mean none were there. After all, if someone as careful as Frege had allowed contradiction to slip in with the seemingly innocuous Basic Law V, how could anyone's preferred logical system to be safe?

But after 2000 pages and 3 volumes, the authors confessed intellectual exhaustion, not finishing their planned 4th volume, let alone a consistency proof of their system. Moreover, they had not even completed the logicist program to their satisfaction, as their system still depended on 3 axioms which were clearly mathematical and not logical in nature: the axiom of reducibility, the axiom of choice, and the axiom

[11]The type theory in *PM* is more complicated, and includes types of n-ary relations for any n with the types of the arguments determined by arbitrary sequences of n types. The fact that types of *sets* are sufficient to define relation types was not known until 1914, when it was first shown that any ordered pair could be expressed as a set [39, p. 224]. A type theory for *functions* was proposed by Alonzo Church in the 1930's, and it is Church's simply typed lambda calculus, one of the (very early) precursors of the Calculus of Constructions discussed in chapter 3.

of infinity. While the former two are beyond our scope here, the axiom of infinity simply asserts that infinite sets exist. Clearly, this was what much of the controversy with actual infinity was about. Feeling unable to answer their critics despite about spending 2 decades of their own lives, they too left the field of mathematical logic.

1.7 Intuitionism

By the year 1920, Cantor and Dedekind were no longer alive, and neither were the most prominent early critics of set theory, Kronecker and Poincare. With Russell and Whitehead having abandoned their quest as well, the logicist program was no more. But in their place, new battle lines were already being drawn.

In the nearly 50 years since Cantor and Dedekind began their work, mainstream mathematics became even more abstract, and even further removed from its original motivations in the physical sciences. The intuitions of number, space, and time, which had guided mathematical thought from time immemorial, was no longer the North Star it used to be in some of the era's new and exciting results:

Theorem (Brouwer's fixed point theorem)**.** *If $X \subseteq \mathbb{R}^n$ is homeomorphic to the unit ball, then any continuous function $f : X \to X$ has a fixed point $x = f(x) \in X$.*

Theorem (Hilbert's Nullstellensatz)**.** *If I be an ideal over an algebraically closed field K, then for any $p \in K[x_1, ..., x_n]$ that vanishes on $V(I)$, there is some $r \in \mathbb{N}$ such that $p^r \in I$.*

Divorced from any commonsense intuitions and untethered from what any outsider would call "reality", some mathematicians began to wonder whether their Ivory Tower lectures were grounded in *anything* reasonable, or if their field had become

meaningless sophistry. What was needed was a logical grounding of mathematics in some fixed axioms; otherwise, we could simply invent the above theorems along with their "proofs", and this would pass for good mathematics as long as we used the appropriate jargon.

Of course, it was now clear that mathematics could not be done with just *logical* axioms; the logicist program had clearly failed in 5 volumes and 3 nonlogical axioms. But in this new view, having some mathematical axioms was fine, as long as they were *consistent*, so that they prevented an "anything goes" situation where anything was provable. On a different view, the abstruse "theorems" above *are* meaningless; after all, they could no longer claim any connection to the physical intuitions that breathed life into mathematics in the first place. Roughly speaking, these views would mature in the 1920's into the rival schools of *formalism* and *intuitionism* [34], and the clash between them became known as the *Grundlagenstreit* ("foundational dispute").

L.E.J. Brouwer (1881-1966) was a rising star in the mathematical world at this time, recently making a name for himself with his deep results in topology, including his fixed point theorem above. But he was uneasy with the direction his field had taken in past decades–including some of his own work–and in the late 1910's, he began to speak his mind.

According to intuitionism, mathematics is a creation of the human mind, which organically develops ideas one at a time. These ideas spring from physical intuitions, which may change over time as we understand the world around us. Moreover, the fancy symbols that we use to represent math, such as $+, \int, \in, \delta$, are merely tools to communicate these ideas between different minds.

In Brouwer's view, is is inaccurate to say that all mathematical statements are true or false. Rather, there are the statements that have been proved, the statements

that have been refuted, and those which have not yet been *decided*. After all, since mathematics is a mental process, and this process runs as time progresses, it follows that mathematics yesterday is not the same as it is today. Since there are plenty of unsolved problems in math, we cannot always say up front whether some statement A is provable or if A is disprovable.

But this has a radical implication: $A \vee \neg A$ need not hold, because we might not have proved A yet. But $A \vee \neg A$, the simple belief that A is either true or false, is the Law of Excluded Middle (LEM) which had been considered a fundamental principle of logic since the time of Aristotle. In this framework, LEM *does* hold for finite sets, but Brouwer argued that it was a mistake to blindly carry this Law over to infinite sets, since these are unending objects that we will always have incomplete information about, and hence for any amount of time we study them, there will always be statements we have not proved yet.

1.8 Formalism

As the 20th century began, David Hilbert (1862-1943) showed his stature in the mathematical world when he proclaimed 23 problems for mathematicians to work on over the century. Given at the famous 1900 International Congress of Mathematicians, Hilbert's problems, as they came to be known, would come to guide much of the course of 20th century mathematics. Hilbert's name today is attached to even more theorems than he had problems that day, and his eminence as a mathematician was rivalled only by Poincare, and none after the latter's 1912 death.

Hilbert, in contrast to Poincare, was a forceful advocate of Cantor's set theory, and in fact his 1st problem was the continuum hypothesis that Cantor had himself

posed: after famously proving that $|\mathbb{N}| < |\mathbb{R}|$, Cantor asked if there is any set S such that $|\mathbb{N}| < |S| < |\mathbb{R}|$. In asking this, Hilbert was clearly interested in questions arising from set theory, and wanted his fellow mathematicians to take infinite sets as seriously as he did. But the 2nd problem Hilbert described was even more striking (emphasis in original) [14]:

> When we are engaged in investigating the foundations of a science, we must set up a system of axioms which contains an exact and complete description of the relations subsisting between the elementary ideas of that science. The axioms so set up are at the same time the definitions of those elementary ideas; and no statement within the realm of the science whose foundation we are testing is held to be correct unless it can be derived from those axioms by means of a finite number of logical steps...above all I wish to designate the following as the most important among the numerous questions which can be asked with regard to the axioms: *To prove that they are not contradictory, that is, that a definite number of logical steps based upon them can never lead to contradictory results.*

Notably, this was still before Frege's system was shown inconsistent in 1903. Yet even before that fiasco gave impetus to the consistency question, here Hilbert already showed concern to secure the foundations of his field, lest they be pulled out from under him as later befell Frege.

In 1899 Hilbert also gave a rigorous axiomatization of geometry that answered many of the doubts that had sprung up about Euclid's geometry, which was by then seen as sloppy and imprecise. Hilbert attempted to prove the consistency of his own axioms, but only achieved a partial result, and showed that the consistency of

geometry reduced to the consistency of mathematical analysis.[12] Nevertheless, Hilbert was primarily a mathematician rather than a logician, and at first he his students did little to follow up on his 2nd problem. For the time, Hilbert's involvement to the brewing foundational crisis was sporadic and relatively minor.

But when Brouwer began espousing his intuitionist philosophy, he awakened a sleeping giant. Hilbert, who had previously respected the young Brouwer, now saw a threat to the soul of mathematics. From the 1920's on, the most prestigious mathematician in the world saw himself as the defender of the new mathematics with its high abstraction and platonic beauty. To Brouwer's rejection of LEM, Hilbert railed [39, p. 476]:

> Taking the principle of excluded middle from the mathematician would be the same, say, as proscribing the telescope to the astronomer or to the boxer the use of his fists. To prohibit existence statements and the principle of excluded middle is tantamount to relinquishing the science of mathematics altogether.

To those who continued to criticize Cantor's set theory, he defied [15]:

> No one shall drive us out of the paradise which Cantor has created for us.

Yet Hilbert himself, evidently, had his own doubts about the actual infinite. In the same 1926 lecture, "On the Infinite", where he defended Cantor's paradise with biblical language, he also pointed to the paradoxes of set theory such as Russell's, acknowledging that they did not reflect well on the set theory he was defending:

> These contradictions, the so-called paradoxes of set theory, though at first scattered, became progressively more acute and more serious. In

[12] By that time, the latter was more familiar and well-understood by mathematicians.

> particular, a contradiction discovered by Zermelo and Russell had a downright catastrophic effect when it became known throughout the world of mathematics...Admittedly, the present state of affairs where we run up against the paradoxes is intolerable. Just think, the definitions and deductive methods which everyone learns, teaches, and uses in mathematics, the paragon of truth and certitude, lead to absurdities! If mathematical thinking is defective, where are we to find truth and certitude?

Indeed, between his rhetorical flourishes, one notices that Hilbert himself had doubts about the infinite, and whether it even made mathematical sense to talk about. As his biography notes [27], he was naturally inclined to skepticism, and felt uneasy placing blind faith in anything, including such an evidently shaky concept as actual infinity.

On the other hand, Hilbert *was* the magistrate of mathematics, and his own life's work was nothing less than the ushering in of the increasingly abstract and powerful methods that defined this new era of math. On some level, he recognized that these new abstractions were detached from the physical intuitions that made us believe them in the first place. But they were *interesting* nevertheless, interesting enough to fill his own lifetime, and the lifetime of many, many mathematicians, then and since, and no one could take that away from them. *No one* would expel them from Cantor's paradise.

The formalists, as Hilbert's school came to be known, likened mathematics to a game of chess: one is allowed to make moves according to certain specific, *formal* rules, and these rules are specified in advance. The players do not make these moves because they have practical importance outside the board; its just a game, after all. Rather, the game is played because it is *interesting*, and that is reason enough.

More than anything, the formalists wanted their Game of Mathematics to be interesting. But if his mathematics was inconsistent, then it would be "anything goes" and would certainly not be interesting. With this in mind, Hilbert stepped up and put the consistency question center stage. When he had first posed it back in 1900, it can be formulated as follows:

Question (Original Hilbert's 2nd Problem)**.** *Is analysis consistent?*

In mathematics, *analysis*, refers to the study of continuous quantities such as the numbers in $\mathbb{R}$.[13] It includes calculus (or rather, the rigorous underpinnings of calculus) and as such, occupies a central place in the mathematical pantheon. Thus, when Hilbert had reduced the consistency of geometry to the consistency of analysis, this was considered an important step, because analysis was by then a more extensively studied subject than geometry.

Analysis differs from arithmetic in that it is mostly about $\mathbb{R}$, while the latter is about $\mathbb{N}$. As Cantor showed, the first set is bigger than the second (assuming, of course, they both exist). For roughly these reasons, analysis is a more powerful system than arithmetic.

On the other hand, while analysis requires some amount of set theory[14] to rigorously axiomatize, set theory itself was advancing far beyond even the abstract considerations in analysis that prompted Cantor and Dedekind to conceive the field. While Cantor had already developed the theory of sets bigger than $\mathbb{R}$, sets in the 20th century were much, *much* bigger.

[13]However, analysis is somewhat of an umbrella term, and also includes *complex analysis* and *functional analysis*, which are concerned with somewhat more abstract objects.

[14]Or something equivalent, such as the type theory mentioned earlier.

In 1908, responding to the controversy surrounding the *axiom of choice*,[15] Ernst Zermelo (1871-1953) formulated a set of axioms of set theory suitable for axiomatizing in greater generality than before. Moreover, while he was unable to give a full proof of consistency, he showed that his system avoided Russell's paradox.[16] In the 1920's, Abraham Fraenkel and Thoralf Skolem proposed some strengthenings to Zermelo's scheme, resulting in the very powerful theory called ZFC (Zermelo-Fraenkel with Choice) which was capable of axiomatizing *all* of mathematics: arithmetic, analysis, and far, far beyond. Today, ZFC is often taken as *the* axiomatization of mathematics as a whole. It did not occupy this central place until around the 1950's and 1960's, with rival systems such as *PM*'s type theory having supporters, but in the 1920's it was clear that set theory was capable of axiomatizing all of math. But with a system so strong–grander even than the Cantor-Dedekind set theory that already provoked such stinging criticism–it became important to prove that *this* system was consistent, and this became the holy grail of Hilbert's program[17]:

[15]Controversy over this axiom played a secondary role in the foundational disputes over set theory and type theory, though we will not elaborate on this here, but see [1].

[16]Zermelo himself had discovered this paradox slightly before Russell, but evidently did not realize its implications and publish this result. His new system avoided the antinomy using a different strategy than Russell's approach in *PM*.

[17]In the late 1920's, Hilbert's program evolved from being (mostly) about consistency to demanding the stronger notion of *conservativity*. If T is some formal system in the language L, a stronger system T' with a more expressive language L' is *conservative over* T if it does not prove any statement in L that T did not already prove (even though it may be prove new statements in L' that could not be expressed in L). To Hilbert, likely influenced by his top student Hermann Weyl (1885-1955) (who was in turn influenced by Brouwer), this became important when he made the distinction between "real" mathematical statements–those justified at least indirectly by empirical reality and physical intuitions–and the "ideal" statements that Brouwer found meaningless. Whereas in the old math where actual infinity was disallowed, it was only possible to express "real" statements, but in the more powerful language of set theory, it was now possible to both express and prove "ideal" statements. Hilbert wanted a proof that set theory was conservative over the the "real" math of earlier times. This way, whenever set theory was used to prove a "real" statement, it would be known that the statement could also be proven in the more trustworthy "real" mathematics. Since the latter proof would often be more difficult and less elegant, set theory would be seen as a convenient set of tools for proving these "real" statements more easily [44].
It should also be noted that Hilbert's program evolved in many other directions in the late 1920's.

Question (Strong Hilbert's 2nd Problem)**.** *Is set theory consistent?*

But even to Hilbert, this was rather ambitious, and to anyone less entranced by this level of idealism, even a consistency proof for analysis was a rather tall order. For these reasons, mere mortals in the burgeoning field of mathematical logic began to increasingly consider how to prove the consistency even of arithmetic:

Question (Weak Hilbert's 2nd Problem)**.** *Is arithmetic consistent?*

And even this revealed itself to be a difficult problem. But even before mathematicians could realize this, it had to be stated unambiguously just what we mean by "arithmetic." That is, it needed a precise axiomatization.

1.9 Peano Arithmetic

Even before Frege was axiomatizing arithmetic as a stepping stone to his grander logicist program, others had been making progress on this first step of putting the basic concept of the natural numbers under the scope of the axiomatic method. Building on the work of Hermann Grassman and Richard Dedekind, in a landmark 1889 treatise Giuseppe Peano gave the first suitably simple and rigorous set of axioms for arithmetic. The resulting axiomatic theory came to be called Peano arithmetic (PA), and proving the consistency of these axioms came to be seen as fundamental to securing the foundations of the rest of mathematics.[18]

For instance, there was the demand of *completeness*, i.e. of showing that for every A, either A or $\neg A$ is provable. There was *decidability*, the problem of showing that there exists an algorithm that can decide in a finite amount of time whether or not a given A is a theorem of some system. All these questions are deep, and continue to be studied in some form by contemporary research in proof theory. Our aim in this thesis, however, is entirely on the consistency question, which certainly had a central place in Hilbert's program throughout.

[18]The 1889 axioms were given in the language of second-order logic, while the importance of instead using first-order logic was not widely recognized until the 1930's [7]. Following other standard

Like every axiomatic system we discuss henceforth, PA is a theory in the language of FOL and has as its *logical* axioms the 5 axioms listed in section 3. In addition, PA has the function symbols $S, +, \cdot$, the constant symbol 0, and the predicate symbol $=$. In addition to the 5 logical axioms, PA has the following *arithmetical* axioms:

(1)	$(\forall x, y, z)$	$x = y \wedge y = z \rightarrow x = z$
(2)	$(\forall x, y)$	$x = y \rightarrow S(x) = S(y)$
(3)	$(\forall x)$	$S(x) \neq 0$
(4)	$(\forall x, y)$	$S(x) = S(y) \rightarrow x = y$
(5)	$(\forall x)$	$x + 0 = x$
(6)	$(\forall x, y)$	$x + S(y) = S(x + y)$
(7)	$(\forall x)$	$x \cdot 0 = 0$
(8)	$(\forall x, y)$	$x \cdot S(y) = x \cdot y + x$

Finally, we have the axiom *schema* of induction, which gives an infinite number of individual axioms. Namely, for any formula $P(x)$, the following is an axiom of PA:

$$(\mathrm{I}_{P(x)}) \qquad P(0) \wedge \forall x(P(x) \rightarrow P(S(x))) \rightarrow \forall x P(x)$$

In this scheme, axioms (1-2) are the *equality* axioms, (3-4) are the *successor* axioms, (5-6) are the *addition* axioms, and (7-8) are the *multiplication* axioms. Peano

sources, we exclusively discuss the first-order version of PA; though this muddles the historical discussion, we claim this oversimplification does not make the basic themes here suffer *too* much.

arithmetic is strong enough to carry formalize just about any reasoning about the natural numbers that one can imagine. It has long been known that PA gets most of its power from the axiom schema of induction. To understand this power, it helps to consider what one gets *without* induction: consider the theory consisting of only the axioms (1-8) except including the following axiom (9)[19] :

$$(9) \qquad (\forall x) \;\; x = 0 \vee \exists y(x = S(y))$$

This is the well-known theory called Robinson's Q (or simply Q) which is capable of proving any quantifier-free statement involving addition and multiplication, such as $10 \cdot 10 \cdot 10 + 9 \cdot 9 \cdot 9 = 12 \cdot 12 \cdot 12 + 1$. However, it is too weak to prove almost anything nontrivial that involves quantifiers. For instance, it *cannot* prove the commutativity of addition:

$$(\forall x, y) \;\; x + y = y + x$$

Because Q is incapable of formalizing almost any interesting arithmetic, while PA is the opposite, proof theorists often like to study axiomatic theories between them in strength, and often this is done by adding to Q a small amount of PA's induction schema. Notably, rather than adding to Q an induction axiom for *every* formula, we can instead do so for every formula with *bounded quantifiers*. Such a formula $P(x)$ may look like:

$$(\exists y \leq x \cdot x \cdot x) \; y \cdot 3 = x + 5$$

[19]The theory we get from (1-8) but without including (9) is generally considered too weak to be interesting.[37] Axiom (9) is redundant in PA because it is provable using induction.

For technical reasons, these are called Δ_0 formulas, and the resulting extension of Q is called $I\Delta_0$ ("Induction over Δ_0 formulas"). $I\Delta_0$ *can* prove the commutativity of addition, as well as most other basic facts about arithmetic [4, p. 85]. However, it is not nearly the system that PA is, being far too weak to formalize all the arithmetic that mathematicians do. Most notably, while $I\Delta_0$ can prove most facts about addition and multiplication, it cannot say anything interesting about exponentiation (as it turns out, PA can).

PA was and is an elegant set of axioms for capturing the concept of number. The trouble is, it has had its critics. Albeit, significantly fewer people have been skeptical of the consistency to PA than have objected to set theory, but nevertheless it has had its doubters, and if *anyone* does not believe the axioms, then those axioms cannot give us the certitude we would like from mathematical proof.

This doubt becomes unsurprising if one scrutinizes PA's axiom schema of induction over *all* formulas. While inducting over bounded, i.e. Δ_0 formulas is perfectly sensible, doing so over arbitrary formulas involves further assumptions. In particular, if we have a PA formula with an unbounded quantifier such as $\exists x$, this quantifier is ranging over our entire domain, namely the completed infinity $\mathbb{N}$. This "x" that apparently exists is no longer something we can necessarily construct. The viewpoint of *potential* infinity only gives us the number $n+1$ when we have the number n, but a statement like $\exists x$ posits a number "somewhere out there" that we perhaps will never see. If we believe that all of $\mathbb{N}$ is there up front, then this is fine, but otherwise, it is hard to justify unbounded induction.

On the other hand, despite the *wide* variety of mathematicians and philosophers who have studied these issues, *no one* of any philosophical persuasion has maintained serious doubts about $I\Delta_0$ (to our knowledge).

The formalists had asked for a consistency proof for arithmetic to settle any skeptics' doubts once and for all, but we have avoided a crucial question so far: how could that consistency proof be formalized, so that no one could reasonably object? The idea of Munchausen's trilemma was well-understood: any rigorous proof would have to begin with *some* axioms, and those axioms would have to be uncontroversial themselves. To prove the consistency of PA, one could not use the unbounded induction of PA in the first place, because anyone disbelieving in PA would mistrust such a proof in the first place. And one certainly could not use a stronger (and thus even less trustworthy) system like ZFC.

What was needed was a specific system whose consistency no one could doubt. From there, and only from there, could the consistency proof for PA be carried out. And maybe then, the consistency of analysis could be established, and just maybe, with enough zeal, this could be bootstrapped up to set theory, and Cantor's paradise would be vindicated at last. But what was needed was the firm foundation on which to begin this edifice: axiomatic bedrock.

Unfortunately, for all of Hilbert's dedication to precision in the axiomatic method, he never did state unambiguously what he felt this bedrock should be. However, he did emphasize that the techniques used in a consistency proof must be "finitary": justified by our own experience with the numbers we actually encounter in practice, and not appealing to any notion of actual infinity, even in disguised form as with PA. Following the influential analysis of [36], it is often held by contemporary experts that Hilbert's "finitary" methods are one and the same with one specific axiomatic system: primitive recursive arithmetic.[20]

[20] Although this view is not unanimous, and as we note below, there is particular disagreement about whether quantifier-free induction over certain transfinite ordinals was considered permissible by the Hilbert school.

1.10 Primitive Recursive Arithmetic

It is not an accident that *primitive recursive arithmetic* (PRA) has come to be associated with finitary reasoning: the paper where Skolem first defined it was titled (in translation), "The foundations of elementary arithmetic established by means of the recursive mode of thought without the use of apparent variables ranging over infinite domains" [39, p. 302]. To formulate this system, Skolem first had to define a certain set of functions on the natural numbers that he felt were finitary, the *primitive recursive functions*. These include:

i The constant zero function: $f(x) = 0$.

ii The successor function, $S(x)$.

iii The projection functions: for any n and any $i \leq n$, we have the function denoted P_i^n defined by $P_i^n(x_1, ..., x_n) = x_i$.

Besides these, there are two rules for building new primitive recursive functions:

iv Composition: If $f : \mathbb{N}^k \to \mathbb{N}$ is primitive recursive and $g_1, ..., g_k : \mathbb{N}^m \to \mathbb{N}$ are all primitive recursive, then so is the function h defined by $h(x_1, ..., x_m) = f(g_1(x_1, ..., x_m), ..., g_k(x_1, ..., x_m))$.
(e.g. if $k, m = 1$, then this is just $h(x) = f(g(x))$.

v Primitive recursion: If $f : \mathbb{N}^k \to \mathbb{N}$ and $g : \mathbb{N}^{k+2} \to \mathbb{N}$ are both primitive recursive, we can define a new primitive recursive function $h : \mathbb{N}^{k+1} \to \mathbb{N}$ as follows:

$$h(0, x_1, ..., x_k) = f(x_1, ..., x_k)$$
$$h(S(y), x_1, ..., x_k) = g(y, h(y, x_1, ..., x_k), x_1, ..., x_k)$$

For instance, using the primitive recursion rule, we can build the addition function $h(y, x) = y + x$. In the above, let f be P_1^1 and let g be $S \circ P_2^3$ (which is primitive recursive by rule (4)). Then we have:

$$h(0, x) = P_1^1(x) = x$$
$$h(S(y), x) = S(P_2^3(y, h(y, x), x)) = S(h(y, x))$$

as desired. And in fact, we can build multiplication from addition in a similar way, then build exponentiation from multiplication, superexponentiation from exponentiation, etc. Just about any function on the natural numbers we would naturally think of is primitive recursive, and in fact it took a few years before Ackermann devised a function that was *not* primitive recursive.

The first 2 axioms of PRA will be (1-2) in the axioms of $PA/Q/I\Delta_0$ above. PRA also has axioms (3-8), the defining equations for $S, +$, and $\cdot$, but it will have much more than that: the defining equations for *every* primitive recursive function. Finally, it will have induction over *quantifier-free* formulas, i.e. formulas without quantifiers at all, even bounded ones. This is a slightly more restrictive condition than Δ_0. For instance, the Δ_0 formula we gave above as $P(x)$:

$$(\exists y \leq x \cdot x \cdot x)\ y \cdot 3 = x + 5$$

is *not* quantifier-free, and so PRA is not able to use induction on it.[21]

As it turns out, PA is itself able to define *all* primitive recursive functions and

[21]It is possible to formally show that theories with Δ_0 induction *are* (sometimes) stronger than quantifier-free induction. For instance, if we let IE_0 denote the theory Q extended with quantifier-free induction, then the "Tennenbaum phenomenon" does not apply to IE_0 but does apply to $I\Delta_0$ [42].

more, using its induction schema [33]. Proving this very nontrivial fact is beyond our scope here, but the upshot is that $PRA \subseteq PA$, i.e. PRA is weaker.[22]

However, PRA itself can formalize almost all arithmetic that mathematicians are interested in, and the extra strength of PA comes from certain extremely fast-growing functions that are mostly of interest to logicians. Nevertheless, the PA axioms are more elegant, and no doubt it would be inconvenient if number theorists ever had to retreat to PRA, where any induction proof can only be conducted with a quantifier-free formula.

1.11 Paradise Lost

As the 1920's marched forward, Hilbert promoted his 2nd problem with increasing zeal, and results finally began to come from his close collaborators: Paul Bernays, Wilhelm Ackermann, and John von Neumann. Hilbert and Bernays developed a technique called the ϵ-calculus, which Ackermann used in his 1924 dissertation to give a consistency proof, up to Hilbert's standards, of a certain system of analysis,

[22]Our description of PRA here is slightly off: it actually has *no* quantifiers, e.g. instead of axioms (1-8), it has the result of removing the universal quantifiers from those. For instance, $(\forall x)x + 0 = x$ becomes $x+0 = x$. Also, quantifier-free induction is a rule of inference rather than an axiom schema, and has the form:

$$\frac{P(0) \qquad P(x) \to P(x+1)}{P(x)} \text{ Quantifier-free Induction}$$

Any statement of the form $P(x)$ is naturally interpreted as $\forall x P(x)$, so even though the language of PRA does not have the symbols $\forall, \exists$, it can still effectively express $\forall$ while avoiding the potential objections to $\exists$. While finitism objects to $\exists$ for reasons noted above, in this view $\forall x P(x)$ is interpreted as "for every number x that *we can construct*, $P(x)$ holds" and so is unproblematic.
Also, PRA and PA have incomparable languages: only the latter has $\forall$ and $\exists$, while only the former has the function symbols for primitive recursive functions beyond $S, +$, and $\cdot$ (PA can still define all these functions and refer to them for all practical purposes, it just doesn't literally have notation like x^y). Thus when we say $PRA \subseteq PA$ we are being somewhat imprecise, but essentially mean that if PRA proves A, then PA proves A', where A' is the result of translating A into the language of PA.

but this was found to be erroneous shortly thereafter by von Neumann. The latter, for his part, gave a 1927 consistency proof in a different style for a different system, but this did not include the induction schema, which as we have seen, is typically the most difficult part of consistency proofs. Nevertheless, Ackermann in that same year gave a second consistency proof, which Bernays now looked over more carefully and accepted [43]. Optimism was now high in the Hilbert school, and in 1928 Hilbert himself gave a lecture declaring that the consistency of arithmetic had been settled, and analysis was just around the corner [44].

In 1930, the young Kurt Gödel (1906-1978) began to approach these problems independently from this group. By the end of the year, he dropped a bombshell, his infamous 2nd Incompleteness Theorem, which in modern terms, roughly says:

Theorem (Gödel's 2nd Incompleteness Theorem)**.** *Let T be a consistent, computably enumerable theory with $PRA \subseteq T$. Then $T \nvdash Con_T$.*

An axiomatic theory is *computably enumerable*, abbreviated *r.e.*, if its axioms can be computably listed out; generally, non *r.e.* theories are considered uninteresting. We will henceforth use the notation $T \vdash A$ to denote that T proves A, and $T \nvdash A$ to denote that T does not prove A. The formula Con_T is the statement that T is consistent, and this statement can be formalized as an arithmetical formula in strong enough arithmetical systems T. $PRA \subseteq T$ means that T is at least as strong as PRA, i.e. T proves every statement that A proves.[23]

If PA *is* consistent, then it certainly satisfies these conditions (as do most theories we are concerned with), but this means:

Corollary 1. $PA \nvdash Con_{PA}$

[23]Strictly speaking, we do not need T to be as strong as PRA for this theorem to apply, but PRA is a convenient landmark for our purposes.

And since $PRA \subseteq PA$, Gödel's theorem also immediately implies:

Corollary 2. $PRA \nvdash Con_{PA}$

Gödel's results, once they came to be understood and accepted, came to be seen as the death knell of even the weak version of Hilbert's 2nd problem. Ackermann's second attempt at a consistency proof, praised in Hilbert's 1928 lecture, was now scrutinized again and was shown to fall through, the error being found by von Neumann. The latter was legendary for his quickness of mind, and was by far the first to understand Gödel's results and draw out the corollaries. The leading young light of Hilbert's acolytes–and perhaps the mathematical world writ large–he advanced the point that *any* nontrivial consistency proof was now ruled out; for if we did have a rigorous, finitary argument for Con_{PA}, it could be *certainly* be formalized in PA itself. But since Gödel showed PA cannot formalize any such argument, it cannot exist in the first place. Others were slower to comprehend what the 2nd Incompleteness theorem implied, but no one seemed capable of formulating a cogent response.

It was in this intellectual milieu that Gerhard Gentzen (1909-1945) came of age as a logician. Like most, he was initially unsure what to make of Gödel's results, let alone how to proceed in light of von Neumann's stinging but important observations. But he continued to wonder why the consistency of arithmetic felt intuitively obvious, yet this feeling could not be put into mathematical words, or at least, not as a theorem in PA itself. His thought process is observable from his notes, where, noticing that Gödel's results imply PA cannot conclude consistency so easily, asked [40, p. 125]:

Where is the Gödel-point hiding?

In the next few years, while most others had given up once and for all, Gentzen would continue searching for his Gödel-point. Some say he found it, while others remain less sanguine.

1.12 Ordinals

As a foundation for mathematics, set theory is powerful because it is possible to rigorously define any mathematical object as a set. For instance, the natural numbers can be encoded this way, if we let 0 be $\{\}$, the empty set, which is typically denoted by $\emptyset$. Then any $n+1$ can be written as $\{0, 1, .., n\}$, giving us all the natural numbers as follows:

$$
\begin{aligned}
0 &= \emptyset \\
1 &= \{0\} \\
2 &= \{0, 1\} \\
3 &= \{0, 1, 2\} \\
4 &= \{0, 1, 2, 3\} \\
&\vdots
\end{aligned}
$$

In this way, every number is *literally* the set of all numbers smaller than it. Under this definition set of all natural numbers, i.e. $\{0, 1, 2, ...\}$, can almost be regarded as a number bigger than any n, since it contains every n. Cantor took this idea seriously, called this new "number" ω, and imagined continuing to count higher by putting ω in another set:

$$\begin{aligned}
\omega &= \{0, 1, 2, ...\} \\
\omega + 1 &= \{0, 1, 2, ..., \omega\} \\
\omega + 2 &= \{0, 1, 2, ..., \omega, \omega + 1\} \\
\omega + 3 &= \{0, 1, 2, ..., \omega, \omega + 1, \omega + 2\} \\
\omega + 4 &= \{0, 1, 2, ..., \omega, \omega + 1, \omega + 2, \omega + 3\} \\
&\vdots
\end{aligned}$$

And so *ordinals* were born, and a question that arises is, what happens if we continue *this* sequence infinitely? To Cantor, the answer was simple; we just get to:

$$\begin{aligned}
&\vdots \\
\omega + \omega &= \{0, 1, 2, ..., \omega, \omega + 1, \omega + 2, ...\} \\
\omega + \omega + 1 &= \{0, 1, 2, ..., \omega, \omega + 1, \omega + 2, ...\omega + \omega\} \\
\omega + \omega + 2 &= \{0, 1, 2, ..., \omega, \omega + 1, \omega + 2, ...\omega + \omega\,\omega + \omega\ + 1\} \\
\omega + \omega + 3 &= \{0, 1, 2, ..., \omega, \omega + 1, \omega + 2, ..., \omega + \omega\,\omega + \omega\ + 1, \omega + \omega\ + 2\} \\
&\vdots
\end{aligned}$$

and continue on as before. Since he was treating these as (generalizations of) number anyways, Cantor decided to denote $\omega + \omega$ as $\omega \cdot 2$. With this notation, we can more clearly write out this whole sequence so far:

$$0, 1, 2, ...\omega, \omega + 1, \omega + 2, ..., \omega \cdot 2, \omega \cdot 2 + 1, \omega \cdot 2 + 2, ..., \omega \cdot 3, \omega \cdot 3 + 1, \omega \cdot 3 + 2, ...$$

and be tempted to take it even further. But now we can notice that we can isolate

out the subsequence:

$$\omega, \omega \cdot 2, \omega \cdot 3, \omega \cdot 4, \omega \cdot 5, ...$$

and ask where this leads. Naturally, Cantor had this keep going to $\omega \cdot \omega$, which he called ω^2. Noticing that he could make this whole sequence all over again on top of ω^2 to get ω^3, and then do that *again* to get ω^4, Cantor formed the sequence:

$$\omega, \omega^2, \omega^3, \omega^4, \omega^5, ..., \omega^\omega$$

This whole process of reaching ω^ω is best visualized in Figure 1.1 below.

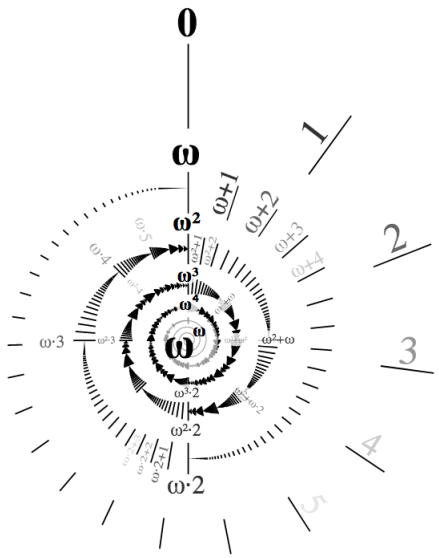


Figure 1.1: The ordinals up to ω^ω

There is, however, no reason to stop there, and Cantor certainly did not. We can run this whole process again to get $\omega^\omega \cdot 2$, and then $\omega^\omega \cdot 3$, and so on:

$$\omega^{\omega}, \omega^{\omega} \cdot 2, \omega^{\omega} \cdot 3, \omega^{\omega} \cdot 4, ..., \omega^{\omega} \cdot \omega$$

and in Cantor's scheme, many (but not all) of the usual rules of exponentiation carry over to from the familiar finite numbers to these infinite ordinals. In particular, this last ordinal here is equivalent to $\omega^{\omega+1}$. Continuing the pattern, we get:

$$\omega^{\omega+1}, \omega^{\omega+2}, \omega^{\omega+3}, ..., \omega^{\omega \cdot 2}, ..., \omega^{\omega \cdot 3}, ..., \omega^{\omega \cdot 4}, ..., \omega^{\omega \cdot \omega}$$

where *this* last term is equal to ω^{ω^2}, and so its natural to continue with:

$$\omega^{\omega}, \omega^{\omega^2}, \omega^{\omega^3}, \omega^{\omega^4}, ..., \omega^{\omega^{\omega}}$$

Finally, abstracting this process even further, we can imagine building:

$$\omega^{\omega}, \omega^{\omega^{\omega}}, \omega^{\omega^{\omega^{\omega}}}, \omega^{\omega^{\omega^{\omega^{\omega}}}}, ...$$

And the question arises as to what ordinal this sequence takes us to. This is the ordinal ϵ_0, which is equivalently defined as the smallest ordinal α such that $\omega^{\alpha} = \alpha$. Every ordinal below ϵ_0 can be written with a finite number of symbols $\{0, 1, 2, ..., 9, \omega\}$, e.g.:

$$\omega^{\omega^{\omega^4 \cdot 6 + \omega + 3} \cdot 2 + \omega^5 + 4} \cdot 3 + \omega^{\omega^{\omega}} \cdot 5 + 11 \qquad (*)$$

What Gentzen ultimately showed was that if one strengthens PRA with the axiom schema of quantifier-free induction over ϵ_0, then this new system proves the consistency of PA. We will call this system $PRA + \epsilon_0$.

This result does not contradict Gödel's theorem because $PRA + \epsilon_0$ is evidently incomparable with PA, since PA seems unable to itself prove this schema. On the other hand, $PRA + \epsilon_0$ does not contain PA,[24] so this result is nontrivial since one could conceivably believe in $PRA + \epsilon_0$ without already trusting PA in the first place.

While the ordinals discussed above naturally seem infinitary in character, they can in fact be coded into arithmetical theories as certain combinatorial objects. For instance, in this way, PA actually proves quantifier-free induction over ω^ω, which is enough to prove the consistency of PRA. In fact, Gentzen's theorem also opened up the field of ordinal analysis, where different theories can be calibrated in strength by assigning them a *proof-theoretic ordinal*, which, roughly speaking, is the least ordinal α such that quantifier-free induction up to α is sufficient to prove the consistency of the given theory [26]. For instance, in modern terminology we say that Gentzen showed that PA has proof-theoretic ordinal ϵ_0, while PRA and $I\Delta_0$ have proof-theoretic ordinals ω^ω and ω^2, respectively.

Finally, it is important to note that only quantifier-free induction is used, and so every formula inducted on in Gentzen's proof is itself finitary in character. After all, if we had unbounded induction, then we would already have PA, since it gets its strength merely from unbounded induction over ω.

1.13 What is ϵ_0?

Up to now, we have not attempted to convince the skeptical reader of the validity of quantifier-free induction over ϵ_0, but it is known to have a finitary interpreta-

[24] The proof of this claim is nontrivial, but see [29].

tion. In fact, Ackermann's original 1924 dissertation under Hilbert's supervision used induction over ω^{ω^ω} [43].

For instance, ϵ_0 can be viewed as an ordering on finite rooted trees [16]. It is also an ordering on *hereditarily finite* lists: finite lists, finite lists of finite lists, finite lists of finite lists of finite lists, etc. For excellent elaborations of this, we recommend [35] and [5] (the latter, in particular, is the best contemporary exposition of the consistency of PA that we know of).

We don't care about the elements of these lists, just their "membership structure". In this way, we can write them using only (matching) parentheses, as illustrated in the table given on the next page.

In this scheme, ordinal addition is (usually) given by concatenating our lists, e.g. $\omega + 2$ is given by concatenating (()) and ()(), yielding (())()(). The exponential ω^α is given by enclosing α's list in parentheses, e.g. $\omega^{\omega+2}$ is ((())()()). From these two facts, it is possible to define ordinal multiplication, and hence all of ordinal arithmetic. We encourage the reader to attempt to write down the corresponding list for the ordinal (*) given on page 39.

We say "usually" in the previous paragraph because, to add ordinals in that way, we need them to be in *Cantor normal form*: in ordinal arithmetic, $1 + \omega = \omega$ (even though $\omega + 1 \neq \omega$), so $1 + \omega$ is not in normal form until we rewrite it as ω. By the same token, ()(()) = (()) in our scheme.

When we assert that we can induct over ϵ_0, we are claiming that this ordering is *well-founded*: there is no infinite descending chain of ordinals

$$\epsilon_0 > \alpha_1 > \alpha_2 > \alpha_3 > ...$$

Just as there is (apparently) no sequence of numbers that descend:

$$\omega > n_1 > n_2 > n_3 > ...$$

Of course, we are only asking for *quantifier-free* induction over ϵ_0, meaning that the sequence of α_i's have to be picked out by a quantifier-free (and hence finitary) formula. It may be useful to call this condition *weak well-foundedness*.

To be truly convinced that we are describing an actual (well-defined) ordering, let alone a weakly well-founded ordering, one has to be able to say *precisely* what this ordering *is*. It turns out that this is rather difficult, as we discuss in section 3.2.

1	()
2	()()
3	()()()
⋮	⋮
ω	(())
$\omega + 1$	(())()
$\omega + 2$	(())()()
⋮	⋮
$\omega \cdot 2$	(())(())
$\omega \cdot 3$	(())(())(())
⋮	⋮
ω^2	(()())
ω^3	(()()())
⋮	⋮
ω^ω	((()))
$\omega^{\omega+1}$	((())())
$\omega^{\omega \cdot 2}$	((())(()))
ω^{ω^2}	((()()))
ω^{ω^ω}	(((())))
⋮	⋮

CHAPTER 2

GENTZEN'S CONSISTENCY PROOF

Here we describe Schütte's [28] 1950 reformulation of Gentzen's proof, and our exposition will itself follow that given in the appendix of the first edition of Mendelson's 1964 *Introduction to Mathematical Logic* [20]. The appendix has been admired for being one of the most accessible presentations of Gentzen's difficult theorem, yet it was taken out of later editions of his textbook. When asked why, he replied [32]:

> I omitted it in later editions because I felt that the topic needed a much more thorough treatment than what I had given, a treatment that would require more space than would be appropriate in an introduction to mathematical logic.

Indeed, if one compares it to this chapter, one can find many places where the argument rests on subtleties that are not discussed there, and perhaps would not be properly appreciated by introductory logic students if they were. This becomes even more evident in our Coq implementation of the proof, where we have had to *fully* unwind the details.

In *very* oversimplified terms, the proof will proceed as follows: to prove the consistency of PA, it will be easier to prove the consistency of a certain stronger system called PA_ω, which has different axioms than PA, and many more inference rules. To argue that $0 = 1$ will never be the conclusion of a proof in PA_ω, we will

show that all these inference rules have the *subformula property*, meaning that the premise(s) is (are) a *subformula* (subformulas) of the conclusion. This means that to prove a formula like $0 = 1$, we must start with a subformula of $0 = 1$ and apply our inference rules. But since $0 = 1$ has no subformulas (besides itself), it is not provable.[1]

2.1 The System PA_ω

Any closed atomic formula in our language will be of the form $s = t$, where s and t are terms built up only from the symbols $0, S, +, \cdot$. For instance, $1 + 1 = 2$ and $(5 + 8) \cdot 5 = 4 \cdot 4 \cdot 4$ are possible closed atomic formulas. The former we will call *correct*, because when we evaluate the operations on either side of the equality, we end up with the same term on both sides. The latter we will call *incorrect*, because the evaluation yields $65 = 64$, and 65 and 64 are clearly different terms. Note that this evaluation process can always be completed in finite time, i.e. is computable.

The axioms of PA_ω consist of:

- All correct closed atomic formulas
- The negations of all incorrect closed atomic formulas

So $1 + 1 = 2$ and $\neg((5 + 8) \cdot 5 = 4 \cdot 4 \cdot 4)$ are axioms of PA_ω. Thus, just from its axioms, PA_ω "knows" everything about quantifier-free statements of arithmetic, but nothing about any quantified statements like $\forall n : n + 0 = 0$ or $\exists n : S(n) = 0$. These will be handled by its rules of inference, which we classify as either *weak*, *strong*, or *Cut*:

[1] This paragraph is simply meant to provide intuitions; in the actual proof, we will not actually use the concept of subformulas, and so we will not define it precisely.

1. Weak Rules:

   (a) Exchange:

$$\frac{C \vee A \vee B \vee D}{C \vee B \vee A \vee D}$$

   (b) Contraction:

$$\frac{A \vee A \vee D}{A \vee D}$$

2. Strong Rules:

   (a) Weakening:

$$\frac{D}{A \vee D}$$

   (b) DeMorgan:

$$\frac{\neg A \vee D \qquad \neg B \vee D}{\neg(A \vee B) \vee D}$$

   (c) Negation:

$$\frac{A \vee D}{\neg\neg A \vee D}$$

   (d) Quantification:

$$\frac{\neg A(t) \vee D}{\neg(\forall x A(x)) \vee D}$$

(e) ω-Rule:

$$\frac{A(n) \vee D \text{ for each } n \in \mathbb{N}}{(\forall x A(x)) \vee D}$$

3. Cut:

$$\frac{C \vee A \qquad \neg A \vee D}{C \vee D}$$

In all these rules, C and D will be called the *side formulas*, and are optional, except that D must occur in Weakening and at least one of C, D must occur in Cut. For instance, Exchange gives us four rules:

1. Conclude $C \vee B \vee A \vee D$ from the premise $C \vee A \vee B \vee D$
2. Conclude $B \vee A \vee D$ from the premise $A \vee B \vee D$
3. Conclude $C \vee B \vee A$ from the premise $C \vee A \vee B$
4. Conclude $B \vee A$ from the premise $A \vee B$

We will later prove that these rules make PA_ω a strictly stronger system than PA. Of particular note is that the induction schema of PA is subsumed by our much stronger ω-Rule. Note that the ω-Rule completely upends our proof system, since it requires infinitely many premises. Anywhere else in logic, a rule is something a finite reasoner can use, e.g. "given premises $P_1, P_2, ..., P_n$, infer conclusion C". We can imagine the ω-Rule as requiring a function f_A where for each $n \in \mathbb{N}$, $f_A(n)$ is a proof of $A(n)$.

To see that the ω-Rule can prove anything regular induction can, suppose we can prove both $A(0)$ and $\forall n(A(n) \to A(n+1))$. Then we can let $f_A(0)$ be our proof of

$A(0)$, and if $f_A(k)$ has been defined, we can let $f_A(k+1)$ be the composition of $f_A(k)$ and $A(k) \to A(k+1)$. This gives us our function f_A, which returns a proof of $A(n)$ for any n, and hence the ω-Rule allows us to conclude $\forall n A(n)$.

In fact, the ω-Rule is much stronger than regular induction; for instance, it can prove $Con(PA)$. Since $PA_\omega \vdash Con(PA, n)$ for each concrete natural number n (for the same reasons that PA proves these statements), PA_ω can then infer $\forall n Con(PA, n)$, i.e. $Con(PA)$.

2.2 Outline of the Consistency Proof

Of course, our point is not that $PA_\omega \vdash Con(PA)$, but to show that $PRA + \epsilon_0 \vdash Con(PA)$. Since PA_ω is stronger than PA, it will suffice to show $PRA + \epsilon_0 \vdash Con(PA_\omega)$. As it turns out, despite the fact that PA_ω is stronger than PA, and even though the ω-Rule, on its face, makes our proof system more complicated, it turns out that this exact move will make it possible to reason about proofs, and ultimately, show that none of them terminate with $0 = 1$.

We said earlier that we can't derive $0 = 1$ in PA_ω, we note that $0 = 1$ has no subformulas, so any proof of $0 = 1$ would have to use an inference rule that does *not* have the subformula property.

More accurately, we will show that PA_ω does not prove any statement of the form:

$$0 = 1 \vee 0 = 1 \vee ... \vee 0 = 1$$

Such a formula, consisting of the disjunction of one or more $0 = 1$ atomic formulas, we will call a *dangerous disjunction*, or simply *dangerous*.[2] None of our axioms are

[2]Technically, we will also consider $0 = 1 \vee (0 = 1 \vee 0 = 1)$ dangerous, as well as any other

dangerous, since none of our axioms are $0 = 1$, and none are disjunctions of two or more formulas. So, if we have imagine there were some derivation in PA_ω that began with axioms and ended in such a contradiction, it would have had to "become dangerous" at some specific step, and so we can ask what this step was. More precisely:

> *What rules of inference can potentially begin with a non-dangerous formula, and conclude with a dangerous formula?*

Such a rule of inference will be deemed *dangerous*, otherwise, *safe*.

Proposition 1. *The only dangerous rule in PA_ω is Cut.*

Proof. Inspecting the rules of inference, the conclusions in DeMorgan, Negation, Quantification, and the ω-Rule cannot be dangerous, since their first disjunct is non-atomic, so these rules are safe.

However, Weakening and any of the Weak Rules do yield disjunctions in their conclusions. But in each of these cases, if the conclusion is a dangerous disjunction, the premise must also be dangerous. For instance, if we used Weakening to conclude $0 = 1 \vee 0 = 1$, then we began with $0 = 1$, which was already dangerous, so it "wasn't Weakening's fault" that we got into danger, and thus these rules are also safe.

On the other hand, Cut can be dangerous, if we take $C :\equiv D :\equiv 0 = 1$, and let A be anything. Then the conclusion, $0 = 1 \vee 0 = 1$, is dangerous, but the right premise, $\neg A \vee 0 = 1$, is *not* dangerous, and neither is the left premise, so in this case we could potentially move into danger. □

Thus, Cut is the only rule that can get us into danger, so any dangerous derivation in PA_ω from its axioms *must* invoke the Cut rule.

disjunction of $0 = 1$ formulas, no matter how the parentheses are grouped.

And what exactly is the Cut rule? To gain a better understanding of what the rule says, it will be helpful to rewrite it, using the equivalence of $\not P \vee Q$ and $P \to Q$:

$$\frac{\neg C \to A \qquad A \to D}{\neg C \to D}$$

Taking $\Gamma :\equiv \neg C$, we obtain[3]:

$$\frac{\Gamma \to A \qquad A \to D}{\Gamma \to D}$$

Thus, Cut simply says that if from some premise(s) Γ we can prove some intermediate result A, and from A we can derive some conclusion D, then we can infer that conclusion directly from our premise(s) Γ. Most mathematicians, of course, assume this all the time, by breaking the proof of some theorem into multiple steps, e.g. with arguments like "first, we will prove Lemma A_1, use it to prove A_2,..., and then invoke Lemma A_n a few times to finally prove our theorem." If Cut were disallowed, mathematicians would not be able to piece together sub-proofs in this way, and have to approach the proof of D in a more roundabout way.

What we will claim, however, is that the use of intermediate results is merely a *convenience* for finishing proofs more efficiently, and that any proof can in principle be done without this convenience. More precisely, any derivation in PA_ω that uses Cut can be done without using Cut, although the Cut-free proof may be much, much longer. For our purposes, this means that if we could prove a dangerous disjunct in PA_ω, then we could make this dangerous derivation *without* using Cut. But since all of the non-Cut rules are safe, this is impossible, so PA_ω can't prove a dangerous disjunct at all.

[3]This is the way the Cut rule is usually formulated in Sequent Calculus

Thus, we must prove two claims:

Claim. *If some formula A is provable in PA, it is provable in PA_ω.*

Claim. *If some formula A is provable in PA_ω, then it can be proved in PA_ω using only the safe rules.*

From which it follows that PA does not prove any dangerous disjunct A, since PA_ω will never prove A using only its safe rules.

2.3 PA $\subseteq$ PA$_\omega$

In this section, we will prove that PA_ω is at least as strong as PA, so that for any formula A, if $PA \vdash A$, then we also have $PA_\omega \vdash A$. Taking $A :\equiv 0 = 1$, it follows that if $PA \vdash 0 = 1$, then $PA_\omega \vdash 0 = 1$, so if PA_ω does not prove $0 = 1$, neither does PA.

(Technically, we will show something slightly weaker, since PA proves non-closed formulas while PA_ω only proves closed formulas. In this section, we will show that if $PA \vdash A$, then PA_ω proves every *closed instance* of A, by which we mean anything we can get by replacing A's free variables with closed terms. If A is closed, then its only closed instance is itself, from which it follows that PA_ω proves every *closed* formula that PA proves. Consequently, the implications for consistency will still apply, and we will only have to manage this technicality about closed instances in part (4) of Lemma 2 and part (2) of Lemma (5).)

Before showing that $PA \subseteq PA_\omega$, it will be helpful to prove some auxiliary lemmas about PA_ω: namely, that it proves associativity of disjunction, $\neg A \vee A$ for any formula A (LEM), and that $s = t \to (A(s) \to A(t))$.

Proposition 2. *The associative rules:*

$$\frac{(C \vee A) \vee B}{C \vee (A \vee B)}$$

$$\frac{C \vee (A \vee B)}{(C \vee A) \vee B}$$

are derivable from 3 applications of the exchange rule. Hence, we can treat Associativity as an additional derived weak rule to abbreviate proofs.

Proof. The exchange rule has the following two special cases, corresponding to when the left or right side formulas are absent:

$$\frac{(A \vee B) \vee C}{(B \vee A) \vee C} \text{ and } \frac{(C \vee A) \vee B}{(C \vee B) \vee A}$$

We use those special cases of the exchange rule to make the following derivations:

$$\frac{\dfrac{\dfrac{(C \vee A) \vee B}{(A \vee C) \vee B}}{(A \vee B) \vee C}}{C \vee (A \vee B)} \text{ and } \frac{\dfrac{\dfrac{C \vee (A \vee B)}{(A \vee B) \vee C}}{(A \vee C) \vee B}}{(C \vee A) \vee B}$$

□

Lemma 1. *For any closed terms s and t, if $s = t$ is correct and $A(x)$ is a formula with x the only free variable (or A is closed), then $PA_\omega \vdash \neg A(s) \vee A(t)$.*

Proof. By induction on n, the number of connectives and quantifiers in A.

$\underline{A \text{ is atomic } (n = 0)}$: Then $A(s)$ is either correct or incorrect, so either $A(s)$ or $\neg A(s)$ is an axiom of PA_ω. If the latter, then we have:

$$\dfrac{\dfrac{\dfrac{\neg A(s)}{A(t) \vee \neg A(s)}\,\text{Weakening}}{\neg A(s) \vee A(t)}\,\text{Exchange}}$$

On the other hand, if $A(s)$ is an axiom of PA_ω, then we claim $A(t)$ also is. This is because $s = t$ is correct, so the terms s, t evaluate to the same value, and hence for any term T and free variable x, the resulting substitutions $T[s/x]$ and $T[t/x]$ evaluate to the same value (this must be shown by induction on terms; we leave the details to our Coq implementation). As an atomic formula, A is of the form $t_1 = t_2$ for some terms t_1, t_2, and since $A(s) \equiv t_1[s/x] = t_2[s/x]$ is correct, so is $A(t) \equiv t_1[t/x] = t_2[t/x]$

A is not atomic: For the inductive step, suppose the claim holds for all $k < n$. We have 3 cases:

1. A is $B \vee C$: By the induction hypothesis, we have proofs of $\neg B(s) \vee B(t)$ and $\neg C(s) \vee C(t)$:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\vdots}{\neg B(s) \vee B(t)}\,\text{Induction Hypothesis}}{C(t) \vee (\neg B(s) \vee B(t))}\,\text{Weakening}}{\neg B(s) \vee B(t) \vee C(t)}\,\text{Exchange}}{\neg B(s) \vee (B(t) \vee C(t))}\,\text{Associativity} \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\vdots}{\neg C(s) \vee C(t)}\,\text{Induction Hypothesis}}{B(t) \vee (\neg C(s) \vee C(t))}\,\text{Weakening}}{\neg C(s) \vee C(t) \vee B(t)}\,\text{Exchange}}{\neg C(s) \vee B(t) \vee C(t)}\,\text{Exchange}}{\neg C(s) \vee (B(t) \vee C(t))}\,\text{Associativity}}{\neg(B(s) \vee C(s)) \vee (B(t) \vee C(t))}\,\text{DeMorgan}$$

2. A is $\neg B$: Then since $t = s$ is correct, by the induction hypothesis $PA_\omega \vdash \neg B(t) \vee B(s)$, so we can derive:

$$\dfrac{\dfrac{\dfrac{\dfrac{\vdots}{\neg B(t) \vee B(s)}\,\text{Induction Hypothesis}}{B(s) \vee \neg B(t)}\,\text{Exchange}}{\neg\neg B(s) \vee \neg B(t)}\,\text{Negation}}$$

3. A is $\forall y B(x, y)$:

$$\dfrac{\vdots \quad \dfrac{\dfrac{\dfrac{\vdots}{\neg B(s,m) \vee B(t,m)}\,\text{Induction Hypothesis}}{\neg \forall y B(s,y) \vee B(t,m)}\,\text{Quantification}}{B(t,m) \vee \neg \forall y B(s,y)}\,\text{Exchange} \quad \vdots}{\dfrac{\forall y B(t,y) \vee \neg \forall y B(s,y)}{\neg \forall y B(s,y) \vee \forall y B(t,y)}\,\text{Exchange}}\,\omega\text{-Rule}$$

Where we've applied the ω-Rule to the statements:

$$B(t,0) \vee \neg \forall y B(s,y), B(t,1) \vee \neg \forall y B(s,y), ..., B(t,m) \vee \neg \forall y B(s,y), ...$$

which are all provable.

□

Corollary 3. *If A is a closed formula, then $PA_\omega \vdash \neg A \vee A$.*

Proof. Take any correct formula e.g. $0 = 0$. If A is closed, $A(0)$ is A, so since $PA_\omega \vdash \neg A(0) \vee A(0)$ by the last lemma, $PA_\omega \vdash \neg A \vee A$. □

We are now ready to show:

Claim. *Any formula provable in PA is also provable in PA_ω.*

We must show that PA_ω proves every axiom of PA, and that for PA's 2 rules of inference (modus ponens and universal generalization), if PA_ω proves the premise then it also proves the conclusion. Recall that PA has 5 logical axioms, 8 arithmetic axioms, and the axiom schema of induction. We first consider the logical axioms:

Lemma 2. *PA_ω can prove all (closed instances of) the 5 logical axioms of PA.*

Proof. 1. $B \to (C \to B)$, i.e. $\neg B \vee (\neg C \vee B)$.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\vdots}{\neg B \vee B}\,\text{LEM Lemma}}{\neg C \vee (\neg B \vee B)}\,\text{Weakening}}{\neg C \vee \neg B \vee B}\,\text{Associativity}}{\neg C \vee B \vee \neg B}\,\text{Exchange}}{\neg B \vee (\neg C \vee B)}\,\text{Exchange}$$

2. $(B \to (C \to D)) \to ((B \to C) \to (B \to D))$, i.e.

$$\neg(\neg B \vee (\neg C \vee D)) \vee (\neg(\neg B \vee C) \vee (\neg B \vee D))$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\vdots}{\neg(\neg B \vee C) \vee (\neg B \vee C)}\,\text{LEM}}{\neg(\neg B \vee C) \vee \neg B \vee C}\,\text{Associativity}}{\neg B \vee \neg(\neg B \vee C) \vee C}\,\text{Exchange} \quad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\vdots}{\neg(\neg B \vee \neg C \vee D) \vee (\neg B \vee \neg C \vee D)}\,\text{LEM}}{\neg B \vee \neg C \vee D \vee \neg(\neg B \vee \neg C \vee D)}\,\text{Exchange}}{\neg B \vee \neg C \vee (D \vee \neg(\neg B \vee \neg C \vee D))}\,\text{Associativity}}{D \vee \neg(\neg B \vee \neg C \vee D) \vee (\neg B \vee \neg C)}\,\text{Exchange}}{D \vee \neg(\neg B \vee \neg C \vee D) \vee \neg B \vee \neg C}\,\text{Associativity}}{\neg C \vee (D \vee \neg(\neg B \vee \neg C \vee D) \vee \neg B)}\,\text{Exchange}}{\neg B \vee \neg(\neg B \vee C) \vee (D \vee \neg(\neg B \vee \neg C \vee D) \vee \neg B)}\,\text{Cut}}{\neg B \vee \neg(\neg B \vee C) \vee (D \vee \neg(\neg B \vee \neg C \vee D)) \vee \neg B}\,\text{Associativity}}{\neg B \vee \neg(\neg B \vee C) \vee \neg B \vee (D \vee \neg(\neg B \vee \neg C \vee D))}\,\text{Exchange}}{\neg B \vee \neg B \vee \neg(\neg B \vee C) \vee (D \vee \neg(\neg B \vee \neg C \vee D))}\,\text{Exchange}}{\neg B \vee \neg(\neg B \vee C) \vee (D \vee \neg(\neg B \vee \neg C \vee D))}\,\text{Contraction}}{\neg B \vee \neg(\neg B \vee C) \vee D \vee \neg(\neg B \vee \neg C \vee D)}\,\text{Associativity}}{\neg(\neg B \vee (\neg C \vee D)) \vee \neg(\neg B \vee C) \vee \neg B \vee D}\,\text{Exchange (4 times)}}{\neg(\neg B \vee (\neg C \vee D)) \vee (\neg(\neg B \vee C) \vee (\neg B \vee D))}\,\text{Associativity (twice)}$$

3. $(\neg B \to \neg C) \to ((\neg B \to C) \to B)$, i.e. $\neg(\neg\neg B \vee \neg C) \vee (\neg(\neg\neg B \vee C) \vee B)$

$$
\dfrac{
\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\neg B \vee B}{\neg\neg\neg B \vee B}\;\text{Negation}}{\neg(\neg\neg B \vee C) \vee (\neg\neg\neg B \vee B)}}{\neg\neg\neg B \vee B \vee \neg(\neg\neg B \vee C)}}{\neg\neg\neg B \vee \neg(\neg\neg B \vee C) \vee B}}{\neg\neg\neg B \vee (\neg(\neg\neg B \vee C) \vee B)}}{}
\quad
\dfrac{\dfrac{\dfrac{\dfrac{
\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\vdots}{\neg B \vee B}}{\neg\neg\neg B \vee B}}{\neg\neg C \vee (\neg\neg\neg B \vee B)}}{\neg\neg\neg B \vee B \vee \neg\neg C}}{\neg\neg\neg B \vee (B \vee \neg\neg C)}
\quad
\dfrac{\dfrac{\dfrac{\dfrac{\vdots}{\neg\neg C \vee \neg C}}{B \vee (\neg\neg C \vee \neg C)}\;\text{Weakening}}{B \vee \neg\neg C \vee \neg C}\;\text{Associativity}}{\neg C \vee (B \vee \neg\neg C)}\;\text{Exchange}
}{\neg(\neg\neg B \vee C) \vee \neg(B \vee \neg\neg C)}\;\text{DeMorgan}}{\neg(\neg\neg B \vee C) \vee \neg B \vee \neg\neg C}\;\text{Associativity}}{\neg\neg C \vee (\neg(\neg\neg B \vee C) \vee B)}\;\text{Exchange}}{}
}{\neg(\neg\neg B \vee \neg C) \vee (\neg(\neg\neg B \vee C) \vee B)}\;\text{DeMorgan}
$$

4. $\forall n B(n) \to B(t)$, i.e. $\neg\forall n B(n) \vee B(t)$, where t is closed.

$$
\dfrac{\neg B(t) \vee B(t)}{\neg\forall n B(n) \vee B(t)}\;\text{Quantification}
$$

5. $\forall n(B \to C(n)) \to (B \to \forall n C(n))$, i.e. $\neg\forall n(\neg B \vee C(n)) \vee (\neg B \vee \forall n C(n))$, if B does not have a free occurrence of n.

$$
\dfrac{\dfrac{\dfrac{\vdots \qquad \dfrac{\dfrac{\dfrac{\neg(\neg B \vee C(n)) \vee (\neg B \vee C(n))}{\neg(\neg B \vee C(n)) \vee \neg B \vee C(n)}}{\neg\forall n(\neg B \vee C(n)) \vee \neg B \vee C(n)}\;\text{Quantification}}{C(n) \vee (\neg\forall n(\neg B \vee C(n)) \vee \neg B)} \qquad \vdots}{\forall n C(n) \vee (\neg\forall n(\neg B \vee C(n)) \vee \neg B)}\;\omega\text{-Rule}}{\neg\forall n(\neg B \vee C(n)) \vee \neg B \vee \forall n C(n)}\;\text{Exchange}}{\neg\forall n(\neg B \vee C(n)) \vee (\neg B \vee \forall n C(n))}\;\text{Associativity}
$$

□

Lemma 3. *PA_ω can prove all 8 arithmetical axioms of PA.*

Proof. All 8 arithmetical axioms of PA are universally quantified. Below, we show that they hold for an arbitray term(s). PA_ω does not have universal generalization, but we can apply the ω-Rule to (1-8) below to achieve the universal statement we desire.

1. $t_1 = t_2 \to (t_2 = t_3 \to t_1 = t_3)$, i.e. $t_1 \neq t_2 \vee (t_2 \neq t_3 \vee t_1 = t_3)$

   It is decidable whether the terms t_1, t_2 evaluate to the same number, so we can consider the cases $t_1 = t_2$ or $t_1 \neq t_2$.

   $\underline{t_1 = t_2}$: Then by Lemma 1, $PA_\omega \vdash t_2 \neq t_3 \vee t_1 = t_3$, so by Weakening, $PA_\omega \vdash t_1 \neq t_2 \vee (t_2 \neq t_3 \vee t_1 = t_3)$

   $\underline{t_1 \neq t_2}$: Then $t_1 = t_2$ is incorrect, so $t_1 \neq t_2$ is an axiom of PA_ω, so we have:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{t_1 \neq t_2}{t_2 \neq t_3 \vee t_1 \neq t_2}\,\text{Weakening}}{t_1 = t_3 \vee (t_2 \neq t_3 \vee t_1 \neq t_2)}\,\text{Weakening}}{t_2 \neq t_3 \vee t_1 \neq t_2 \vee t_1 = t_3}\,\text{Exchange}}{t_2 \neq t_3 \vee t_1 = t_3 \vee t_1 \neq t_2}\,\text{Exchange}}{t_1 \neq t_2 \vee (t_2 \neq t_3 \vee t_1 = t_3)}\,\text{Exchange}$$

2. $t_1 = t_2 \to S(t_1) = S(t_2)$, i.e. $t_1 \neq t_2 \vee S(t_1) = S(t_2)$.

   Either $t_1 = t_2$ or $t_1 \neq t_2$. In the former case, $S(t_1)$ and $S(t_2)$ will have the same value, so $S(t_1) = S(t_2)$ will be an axiom, so we get:

$$\dfrac{S(t_1) = S(t_2)}{t_1 \neq t_2 \vee S(t_1) = S(t_2)}\,\text{Weakening}$$

   In the latter case, $t_1 \neq t_2$ is an axiom, so we get:

$$\dfrac{\dfrac{t_1 \neq t_2}{S(t_1) = S(t_2) \vee t_1 \neq t_2}\,\text{Weakening}}{t_1 \neq t_2 \vee S(t_1) = S(t_2)}\,\text{Exchange}$$

3. $0 \neq S(t)$.

   This will be an axiom of PA_ω.

4. $S(t_1) = S(t_2) \to t_1 = t_2$, i.e. $S(t_1) \neq S(t_2) \vee t_1 = t_2$.

   Either $t_1 = t_2$ or $t_1 \neq t_2$. In the former case, we get:

$$\frac{t_1 = t_2}{S(t_1) \neq S(t_2) \vee t_1 = t_2}\ \text{Weakening}$$

In the latter case, $S(t_1)$ and $S(t_2)$ will have different values, so $S(t_1) \neq S(t_2)$ will be an axiom, so we get:

$$\frac{\dfrac{S(t_1) \neq S(t_2)}{t_1 = t_2 \vee S(t_1) \neq S(t_2)}\ \text{Weakening}}{S(t_1) \neq S(t_2) \vee t_1 = t_2}\ \text{Exchange}$$

5. $t + 0 = t$.

   $t + 0$ and t have the same value, so this will be an axiom.

6. $t_1 + S(t_2) = S(t_1 + t_2)$.

   This will be an axiom of PA_ω.

7. $t \cdot 0 = 0$.

   This will be an axiom of PA_ω.

8. $t_1 \cdot S(t_2) = t_1 \cdot t_2 + t_1$.

   This will be an axiom of PA_ω.

□

Lemma 4. *For any formula $B(n)$, PA_ω can prove the axiom schema of induction for $B(n)$.*

Proof. We want to show $B(0) \to (\forall n(B(n) \to B(n+1)) \to \forall n B(n))$, i.e.

$$\neg B(0) \vee (\neg \forall n(\neg B(n) \vee B(n+1)) \vee \forall n B(n))$$

Claim: For any k, PA_ω proves:

$$\neg(\neg B(0) \vee B(1)) \vee ... \vee \neg(\neg B(k-1) \vee B(k)) \vee B(k) \vee \neg B(0)$$

After showing this claim, and letting $C(x)$ be the formula $\neg B(x) \vee B(x+1)$, we can make the following derivation in PA_ω:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\vdots}{\neg C(0) \vee \neg C(1) \vee ... \vee \neg C(k-1) \vee B(k) \vee \neg B(0)}\text{Claim}}{\neg C(0) \vee (\neg C(1) \vee (... \vee (\neg C(k-1) \vee (B(k) \vee \neg B(0)))..)))}\text{Associativity } (k \text{ times})}{\neg \forall n C(n) \vee (\neg C(1) \vee (... \vee (\neg C(k-1) \vee (B(k) \vee \neg B(0)))..)))}\text{Quantification}}{\neg \forall n C(n) \vee \neg C(1) \vee (... \vee (\neg C(k-1) \vee (B(k) \vee \neg B(0)))..)))}\text{Associativity}}{\neg C(1) \vee \neg \forall n C(n) \vee (... \vee (\neg C(k-1) \vee (B(k) \vee \neg B(0)))..)))}\text{Exchange}}{\neg \forall n C(n) \vee \neg \forall n C(n) \vee (... \vee (\neg C(k-1) \vee (B(k) \vee \neg B(0)))..)))}\text{Quantification}}{\neg \forall n C(n) \vee (\neg C(2) \vee ... \vee (\neg C(k-1) \vee (B(k) \vee \neg B(0)))..)))}\text{Contraction}$$

And repeating the last 4 steps $k-1$ times, we get:

$$\dfrac{\dfrac{\dfrac{\vdots}{\neg \forall n C(n) \vee (B(k) \vee \neg B(0))}}{B(k) \vee \neg B(0) \vee \neg \forall n C(n)}\text{Exchange}}{B(k) \vee (\neg B(0) \vee \neg \forall n C(n))}\text{Associativity}$$

And we can derive this last formula for any k, so by the ω-Rule, we get:

$$\dfrac{\dfrac{\dfrac{\vdots}{\forall n B(n) \vee (\neg B(0) \vee \neg \forall n C(n))}\text{Exchange}}{\neg B(0) \vee \neg \forall n C(n) \vee \forall n B(n)}\text{Associativity}}{\neg B(0) \vee (\neg \forall n C(n) \vee \forall n B(n))}$$

$$\dfrac{\vdots \quad \dfrac{\dfrac{\vdots}{B(k) \vee (\neg B(0) \vee \neg \forall n C(n))}}{} \quad \vdots}{}\omega\text{-Rule}$$

as desired. It remains to show the claim, and we will do so by induction on k

k=0: Then this is just $\neg B(0) \lor B(0)$, which is an instance of LEM.

Induction step: We will abbreviate the formula

$$\neg(\neg B(0) \lor B(1)) \lor ... \lor \neg(\neg B(k-1) \lor B(k))$$

with C. Then our induction hypothesis is:

$$C \lor B(k) \lor \neg B(0)$$

And we want to show:

$$C \lor \neg(\neg B(k) \lor B(k+1)) \lor B(k+1) \lor \neg B(0)$$

Using the induction hypothesis, and the fact that we can prove excluded middle for $B(k+1)$, we can make the following derivation:

$$\begin{array}{ll}
 & \vdots \\
\text{Induction Hyp.} & C \lor B(k) \lor \neg B(0) \\
\text{Weakening} & B(k+1) \lor (C \lor B(k) \lor \neg B(0)) \\
\text{Exchange} & C \lor B(k) \lor \neg B(0) \lor B(k+1) \\
\text{Exchange} & C \lor B(k) \lor B(k+1) \lor \neg B(0) \\
\text{Exchange} & C \lor B(k+1) \lor B(k) \lor \neg B(0) \\
\text{Exchange} & C \lor B(k+1) \lor \neg B(0) \lor B(k) \\
\text{Exchange} & B(k) \lor (C \lor B(k+1) \lor \neg B(0)) \\
\text{Negation} & \neg\neg B(k) \lor (C \lor B(k+1) \lor \neg B(0))
\end{array}$$

$$\begin{array}{ll}
\vdots & \\
\neg B(k+1) \lor B(k+1) & \\
C \lor \neg B(0) \lor (\neg B(k+1) \lor B(k+1)) & \text{Weakening} \\
C \lor \neg B(0) \lor \neg B(k+1) \lor B(k+1) & \text{Associativity} \\
C \lor \neg B(0) \lor B(k+1) \lor \neg B(k+1) & \text{Exchange} \\
C \lor B(k+1) \lor \neg B(0) \lor \neg B(k+1) & \text{Exchange} \\
\neg B(k+1) \lor (C \lor B(k+1) \lor \neg B(0)) & \text{Exchange}
\end{array}$$

$$\begin{array}{ll}
\neg(\neg B(k) \lor B(k+1)) \lor (C \lor B(k+1) \lor \neg B(0)) & \text{DeMorgan} \\
C \lor B(k+1) \lor \neg B(0) \lor \neg(\neg B(k) \lor B(k+1)) & \text{Exchange} \\
C \lor B(k+1) \lor \neg(\neg B(k) \lor B(k+1)) \lor \neg B(0) & \text{Exchange} \\
C \lor \neg(\neg B(k) \lor B(k+1)) \lor B(k+1) \lor \neg B(0) & \text{Exchange}
\end{array}$$

as desired.

□

Lastly, we must show that PA_ω subsumes the 2 rules of inference in PA: Modus Ponens and Universal Generalization.

Lemma 5. *1. If $PA_\omega \vdash A$ and $PA_\omega \vdash \neg A \vee B$, then $PA_\omega \vdash B$.*

2. If PA_ω proves every closed instance of A, then PA_ω proves every closed instance of $\forall n A(n)$.

Proof. 1. Suppose $PA_\omega \vdash A$ and $PA_\omega \vdash \neg A \vee B$. Then we can derive B as follows:

$$\dfrac{\dfrac{\dfrac{\dfrac{\vdots}{A}}{B \vee A}\,\text{Weakening} \qquad \dfrac{\vdots}{\neg A \vee B}}{B \vee B}\,\text{Cut}}{B}\,\text{Contraction}$$

2. Suppose PA_ω proves every closed instance of A. Let $\forall n A'(n)$ be a closed instance of $\forall n A(n)$. Then for any n, $A'(n)$ is the result of substituting all of $A(n)$'s free variables for closed terms, so by assumption $PA \vdash A'(n)$. Then we have:

$$\dfrac{A'(0) \qquad A'(1) \qquad A'(2) \qquad A'(3) \qquad \ldots}{\forall n A'(n)}\,\omega\text{-Rule}$$

□

Putting the last 4 lemmas together, we have:

Theorem 1. *Any closed formula provable in PA is also provable in PA_ω.*

Corollary 4. *If PA_ω is consistent, so is PA.*

Now, the issue is to show that PA_ω is consistent.

2.4 Proofs in PA_ω

In the next section, we will give an algorithm to eliminate all Cuts from any proof in PA_ω. However, to show this algorithm terminates, we will need some extra machinery. In this section, we will assign ordinals (below ϵ_0) to proofs, and prove some other technical lemmas we will need for Cut-elimination.

As already noted, in all our inference rules, the optional formulas C, D are called the side formulas. The required formulas A, B we will call the *principal* formulas. In Cut, the principal formula A will be called the *Cut formula*, and the number of connectives and quantifiers in $\neg A$ will be called the *degree* of the Cut.

Furthermore, in any proof $\mathscr{P}$ in PA_ω, the maximum degree of all Cuts in PA_ω will be called the *degree* of $\mathscr{P}$ (if it's Cut-free, it will have degree 0). If $PA_\omega \vdash A$ with a proof of degree $\leq m$, we will write $PA_\omega \vdash_m A$. In eliminating Cuts, our aim will be to show that for any formula A, if $PA_\omega \vdash_{m+1} A$, then $PA_\omega \vdash_m A$. Applying this result $m+1$ times, it will follow that $PA_\omega \vdash_0 A$, so any proof of A can be made Cut-free.

With this in mind, our initial presentation of PA_ω omitted one crucial detail. In using the ω-Rule:

$$\frac{A(n) \vee D \text{ for each } n \in \mathbb{N}}{(\forall x A(x)) \vee D}$$

we require that the sequence of proofs of $A(0) \vee D, A(1) \vee D, A(2) \vee D, ...$ have some uniform bound $M \in \mathbb{N}$ on their degree. In other words, there is some M such that for every n, $PA_\omega \vdash_M A(n) \vee D$. Otherwise, we could apply the ω-Rule to get a proof of infinite degree, which would cause the proof strategy outlined in the above paragraph to fail. Note that this proviso make PA_ω strictly weaker; we leave it to

the reader to verify that every use of the ω-Rule in the previous section is compatible with this proviso.

We will also talk about the *ordinal* assigned to $\mathscr{P}$ in PA_ω, and say $PA_\omega \vdash_m^\alpha A$ if there is a proof of A that has degree $\leq m$ and ordinal $\leq \alpha$. Since proofs are built out of axioms and rules of inference, we will inductively define the ordinal of a proof $\mathscr{P}$ as follows:

1. If $\mathscr{P}$ only consists of an axiom, then $\mathscr{P}$ has ordinal 0.

2. If $\mathscr{P}$ consists of a Weak Rule applied to some proof $\mathscr{P}'$ with ordinal α, then it has ordinal α.

3. If $\mathscr{P}$ consists of a Strong Rule (or Cut) applied to some premise(s), then $\mathscr{P}$ may be assigned any ordinal greater than the ordinals of the proofs of these premises. In particular:

   (a) If $\mathscr{P}$ consists of Weakening, Negation, or Quantification applied to some proof $\mathscr{P}'$ with ordinal α_1, then $\mathscr{P}$ has ordinal $\alpha > \alpha_1$.

   (b) If $\mathscr{P}$ consists of DeMorgan or Cut applied to some proofs $\mathscr{P}_1$ and $\mathscr{P}_2$ with ordinals α_1 and α_2, then $\mathscr{P}$ has ordinal $\alpha > \max\{\alpha_1, \alpha_2\}$.

   (c) If $\mathscr{P}$ consists of the ω-Rule applied to some proofs $\mathscr{P}_0, \mathscr{P}_1, \mathscr{P}_2, ...$ with ordinals $\alpha_0, \alpha_1, \alpha_2, ...$, then it has ordinal $\alpha > \alpha_i$ for every i.

   When we discuss a specific proof $\mathscr{P}$, such as:

$$\dfrac{\dfrac{\vdots}{A}}{\dfrac{B}{\vdots}}\ \text{Rule, } \alpha\text{, m}$$

Rule will denote the rule of inference used to infer formula B from A, while α and m will be the ordinal and degree (respectively) of the proof of B. If we don't care about one or more of these, we will write $-$ or omit the later values. For instance:

$$\dfrac{\vdots}{B}\,{\scriptstyle -,\,\alpha}$$
$$\vdots$$

will indicate that there is a proof of B of ordinal α, but we are not interested in its degree or the rule(s) used to derive B.

Before proceeding to Cut-elimination, we will also need the following lemma:

Lemma 6 (Invertibility Lemma)**.** *The following three rules of PA_ω are invertible (i.e. given the conclusion in the rule, one can prove the premise(s)):*

1. *Negation*
2. *DeMorgan*
3. *ω-Rule*

Moreover, this can be done without a higher ordinal or degree than that of the original proof.

To prove these 3 cases, we will need to define a notion of *formula substitution*: for formulas A, B, C, $\texttt{FormSub}_{B,C}(A)$ is the result scanning A for disjuncts that are identical to B, and replacing them with C. For instance, $\texttt{FormSub}_{B,C}(D \vee (E \vee B))$ is $D \vee (E \vee C)$, but $\texttt{FormSub}_{B,C}(D \vee (E \vee \neg B))$ is $D \vee (E \vee \neg B)$, since B does not

occur in A as a standalone disjunct, but only inside the negation of a disjunct, and hence is not substituted.

From this, we will also need to define a precise notion of *proof substitution*: if $\mathscr{P}$ is a proof of some formula A, then $\texttt{ProofSub}_{B,C}(\mathscr{P})$ is the result of applying $\texttt{FormSub}_{B,C}$ to A, and then recursively applying $\texttt{FormSub}_{B,C}$ to all the formulas in the derivation of A. This is roughly how we will prove Lemma 6: for instance, with Negation, we will take a proof $\mathscr{P}$ of $\neg\neg B \vee D$, and apply $\texttt{ProofSub}_{\neg\neg B,B}$ to it, giving a proof $\mathscr{P}'$ of $B \vee D$. However, there are two complications with applying this procedure naïvely:

a) If D itself contains instances of $\neg\neg B$, such as if D is $\neg\neg B$, Lemma 6 states that from a proof $\mathscr{P}$ of $\neg\neg B \vee \neg\neg B$ we can get a proof of $B \vee \neg\neg B$, but $\texttt{ProofSub}_{\neg\neg B,B}(\mathscr{P})$ instead gives a proof of $B \vee B$. Hence we really just want to apply $\texttt{FormSub}$ to the first $\neg\neg B$ but the second one is not a *substitution target*. This motivates us to define $\texttt{FormSub}_{B,C,\mathbb{I}}$, which is $\texttt{FormSub}_{B,C}$ except only making substitutions as indicated by some particular *substitution indicator* $\mathbb{I}$. Structurally, $\mathbb{I}$ might (for instance) look like 0 (1 (0 0)), to indicate that in a formula of the form $B \vee (C \vee (D \vee E))$, only C is a substitution target, while B, D, and E are to be left alone.

Furthermore, we will need to change $\mathbb{I}$ as we traverse up the proof, since many of the rules will either change the structure of our formula (e.g. Weakening), move our substitution target(s) (e.g. Exchange), or even multiply them (e.g. Contraction). Thus, it needs to be argued on a rule-by-rule basis that we can keep track of our substitution targets, but here we leave the details to the reader.

b) We may run into compatibility issues with some of the inference rules. In our example with $\texttt{ProofSub}_{\neg\neg B,B}(\mathscr{P})$, if part of $\mathscr{P}$ looked like:

$$\dfrac{\dfrac{\dfrac{\vdots}{B \vee E}}{\neg\neg B \vee E}\,\text{Negation}}{\vdots}$$

After substitution it would become:

$$\dfrac{\dfrac{\dfrac{\vdots}{B \vee E}}{B \vee E}\,\text{Negation}}{\vdots}$$

Which is technically not a valid application of Negation. This is easily remedied: instead of applying $\texttt{ProofSub}_{\neg\neg B,B}$ to $\mathscr{P}$ completely, we will make an exception if Negation shows up like this, in which case, instead of substituting we will simply delete the bottom formula from $\mathscr{P}$.

Thus, to prove part (1) of Lemma 6 rigorously, we will describe a transformation operation `DubNegTransf` such that $\texttt{DubNegTransf}(\mathscr{P})$ is a proof of $B \vee D$ if $\mathscr{P}$ is a proof of $\neg\neg B \vee D$. Most of the time, `DubNegTransf` will simply apply $\texttt{FormSub}_{\neg\neg B,B}$ recursively much like $\texttt{ProofSub}_{\neg\neg B,B}$ does, but it will make an exception if it encounters the Negation rule. We are now ready to spell this out:

Proof of Lemma 6. 1. Negation: Suppose there is a proof $\mathscr{P}$ of $\neg\neg B \vee D$ with ordinal α and degree m. Let $\texttt{DubNegTransf}(\mathscr{P})$ be the result of applying $\texttt{ProofSub}_{\neg\neg B,B}$ to $(\mathscr{P})$ with $\neg\neg B$ as the only substitution target (and keeping track of this as it possibly moves around as we traverse up $\mathscr{P}$). However, in the case of Negation, `DubNegTransf` will do something different (if $\neg\neg B$ shows up in the relevant place indicated below). Namely, if $\mathscr{P}$ is (for some subproofs $\mathscr{P}_1, \mathscr{P}_2$ and formula F):

$$\dfrac{\dfrac{\dfrac{\mathscr{P}_1}{B \vee F}\,\text{-},\, < \alpha, \leq m}{\neg\neg B \vee F}\,\text{Negation},\, \leq \alpha, \leq m}{\mathscr{P}_2}$$

Then we will let $\texttt{DubNegTransf}(\mathscr{P})$ be:

$$\dfrac{\dfrac{\mathscr{P}_1{}'}{B \vee F'}\,\text{-}, < \alpha, \leq m}{\mathscr{P}_2{}'}$$

where $\mathscr{P}_1{}', \mathscr{P}_2{}'$ denote the application of `DubNegTransf` to those subproofs, and F' denotes $\texttt{FormSub}_{\neg\neg B, B, \mathbb{I}}(F)$ for the appropriate substitution indicator $\mathbb{I}$ (we will use this notation for the remainder of this proof). The reader can easily verify that this is the only rule that has to be handled as a special case. Hence, $\texttt{DubNegTransf}(\mathscr{P})$ is a valid proof of $B \vee D$ with ordinal and degree no higher than that of $\mathscr{P}$.

2. DeMorgan: If $\mathscr{P}$ is a proof of $\neg(B \vee C) \vee D$ with ordinal α and degree m, we want to get a proof of $\neg B \vee D$ (the case of $\neg C \vee D$ is similar). Then define $\texttt{DeMorganTransf}(\mathscr{P})$ similarly as with (1), except that it will apply $\texttt{ProofSub}_{\neg(B \vee C), \neg B}$, the substitution target is $\neg(B \vee C)$, and the rule that sometimes needs to be handled differently is DeMorgan. If $\mathscr{P}$ is (for some subproofs $\mathscr{P}_1, \mathscr{P}_2, \mathscr{P}_3$ and formula E):

$$\dfrac{\dfrac{\dfrac{\mathscr{P}_1}{\neg B \vee F}\,\text{-}, < \alpha, \leq m \qquad \dfrac{\mathscr{P}_2}{\neg C \vee F}\,\text{-}, < \alpha, \leq m}{\neg(B \vee C) \vee F}\,\text{DeMorgan}, \leq \alpha, \leq m}{\mathscr{P}_3}$$

Then let $\texttt{DeMorganTransf}(\mathscr{P})$ be:

$$\dfrac{\dfrac{\mathscr{P}_1{}'}{\neg B \vee F'}\,\text{-}, < \alpha, \leq m}{\mathscr{P}_3{}'}$$

And the reader can verify that $\texttt{DeMorganTransf}(\mathscr{P})$ is a valid proof of $\neg B \vee D$ with no higher ordinal or degree than that of $\mathscr{P}$.

3. $\underline{\omega\text{-Rule}}$: If $\mathscr{P}$ is a proof of $\forall x B(x) \vee C$ with ordinal α and degree m, and n is arbitrary, we want to get a proof of $B(n) \vee D$. Define $\omega\text{-}\texttt{RuleTransf}(\mathscr{P})$ analogously with (1-2), except it will apply $\texttt{ProofSub}_{\forall x B(x), B(n)}$ and its special case will be the ω-Rule. If $\mathscr{P}$ is (for some subproofs $\mathscr{P}_1, \mathscr{P}_2$ and formula F):

$$\dfrac{\vdots \quad \dfrac{\mathscr{P}_1}{B(n) \vee F}\,\text{-},\, <\alpha, \leq m \quad \vdots}{\dfrac{\forall x B(x) \vee F}{\mathscr{P}_2}}\,\omega\text{-Rule},\, \leq \alpha, \leq m$$

Then let $\omega\text{-}\texttt{RuleTransf}(\mathscr{P})$ be:

$$\dfrac{\dfrac{\mathscr{P}_1{}'}{B(n) \vee F'}\,\text{-},\, <\alpha, \leq m}{\mathscr{P}_2{}'}$$

□

Finally, there is one more lemma we will need in the next section:

Lemma 7 (Erasure Lemma)**.** *Suppose A is one of $B, \neg B$, with B atomic, and A is not an axiom. If we can make the following derivation in PA_ω:*

$$\dfrac{\dfrac{\dfrac{\mathscr{P}_1}{E}\,\text{-},\, \alpha,\, m}{A \vee E}\,\text{-},\, \alpha+1,\, m}{\dfrac{\mathscr{P}_2}{C \vee A}\,\text{-},\, \alpha+\beta+1,\, m}$$

Then we can also derive:

$$\dfrac{\dfrac{\dfrac{\mathscr{P}_3}{E}\,\text{-},\, \alpha,\, m}{\mathscr{P}_2^*}}{C}\,\text{-},\, \alpha+\beta,\, m$$

Proof. Note that A is atomic and not an axiom, it can only arise from an application of Weakening. Let $\mathscr{P}_2^*$ be the result of "erasing" A from every formula in the subproof $\mathscr{P}_2$. This procedure, like `ProofSub`, requires having a particular target formula, but unlike it, we are actually changing the structure of the formula when we delete part of it. This means there are several rules we have to handle differently. For instance, an instance of Exchange would transform from:

$$\dfrac{\dfrac{\dfrac{\vdots}{G \vee A \vee F \vee H}}{G \vee F \vee A \vee H}\,\text{Exchange}}{\vdots}$$

into:

$$\dfrac{\dfrac{\dfrac{\vdots}{G \vee F \vee H}}{G \vee F \vee H}\,\text{Exchange}}{\vdots}$$

In this case, we can simply delete this redundant part of the tree. Contraction poses a similar problem, since we would naïvely change:

$$\dfrac{\dfrac{\dfrac{\vdots}{A \vee A \vee F}}{A \vee F}\,\text{Contraction}}{\vdots}$$

into:

$$\dfrac{\dfrac{\dfrac{\vdots}{F}}{F}\,\text{Contraction}}{\vdots}$$

since if the bottom A is a deletion target, so are both A's above it. But as with Exchange, we can simply delete this part of the tree. □

2.5 Cut-Elimination in PA_ω

It is in the following proof that we must use transfinite induction up to ε_0:

Theorem 2. *If* $PA_\omega \vdash^{\alpha}_{m+1} A$, *then* $PA_\omega \vdash^{2^\alpha}_{m} A$

Proof. By transfinite induction on α.

If $\alpha = 0$, then there are no Cuts so the degree of the proof can't be $m+1$. We will also need $\alpha = 1$ as a base case (in part (c) below). But if $\alpha = 1$, then a Strong Rule was only used once, and if the degree of the proof is nonzero, it must have been Cut. Hence Exchange, Contraction, and Cut were the only rules used, but these rules all require disjuncts as premises, whereas every axiom is either atomic or the negation of an atomic sentence. It follows that no such proof is possible.

For the induction step, assume that for every $\alpha_i < \alpha$:

$$\text{If } PA_\omega \vdash^{\alpha_i}_{m+1} A, \text{ then } PA_\omega \vdash^{2^{\alpha_i}}_{m} A$$

We will search the proof tree level-by-level for the last application of a strong rule or Cut, and we have separate cases depending on the rule. Each of the non-Cut rules proceed similarly, so we will only show the ω-Rule explicitly. In this case, we have:

$$\dfrac{\dfrac{\vdots}{B(0) \vee C}\,\text{-},\ \alpha_0,\ m+1 \qquad \dfrac{\vdots}{B(1) \vee C}\,\text{-},\ \alpha_1,\ m+1 \qquad \vdots}{\forall x B(x) \vee C}\,\omega\text{-Rule},\ \alpha,\ m+1$$

with every $\alpha_i < \alpha$, so by our induction hypothesis, we can transform this into:

$$\dfrac{\dfrac{\vdots}{B(0) \vee C}\,\text{-},\ 2^{\alpha_0},\ m \qquad \dfrac{\vdots}{B(1) \vee C}\,\text{-},\ 2^{\alpha_1},\ m \qquad \vdots}{\forall x B(x) \vee C}\,\text{-},\ 2^{\alpha},\ m$$

since if $\alpha > \alpha_i$ for all i, then $2^\alpha > 2^{\alpha_i}$ for all i.

For Cut, we have:

$$\dfrac{\dfrac{\vdots}{C \vee B}\,\text{-, } \alpha_1, m+1 \qquad \dfrac{\vdots}{\neg B \vee D}\,\text{-, } \alpha_2, m+1}{C \vee D}\,\text{Cut, } \alpha, m+1$$

which by our induction hypothesis, can be transformed into:

$$\dfrac{\dfrac{\vdots}{C \vee B}\,\text{-, } 2^{\alpha_1}, m \qquad \dfrac{\vdots}{\neg B \vee D}\,\text{-, } 2^{\alpha_2}, m}{C \vee D}\,\text{Cut, } 2^\alpha, m+1$$

where B can be atomic, $\neg E$, $E \vee F$, or $\forall x E(x)$, 4 cases that we will consider separately with the aim of reducing the degree of the final Cut from (up to) $m+1$ down to m. In the last 3 cases, this will be achieved by removing one connective or quantifier from B.

1. B is atomic, and we have:

$$\dfrac{\dfrac{\vdots}{C \vee B}\,\text{-, } 2^{\alpha_1}, m \qquad \dfrac{\vdots}{\neg B \vee D}\,\text{-, } 2^{\alpha_2}, m}{C \vee D}\,\text{Cut, } 2^\alpha, m+1$$

since B is atomic, either B or $\neg B$ is an axiom. If $\neg B$ is an axiom, we can ask where B showed up in the proof of $C \vee B$; the only possibility is Weakening, so we have (for some formula E):

$$\dfrac{\dfrac{\dfrac{\vdots}{E}\,\text{-, } < 2^{\alpha_1}, \leq m}{B \vee E}\,\text{Weakening, } \leq 2^{\alpha_1}, \leq m}{\dfrac{\vdots}{C \vee B}\,\text{-, } 2^{\alpha_1}, m}$$

But then, by Lemma 7, we can derive:

$$\dfrac{\dfrac{\dfrac{\dfrac{\vdots}{E}\,\text{-},\, <2^{\alpha_1},\, m}{C}\,\text{-},\, <2^{\alpha_1},\, m}{D \vee C}\,\text{Weakening},\, 2^{\alpha_1},\, m}{C \vee D}\,\text{Exchange},\, 2^{\alpha_1},\, m$$

If B is an axiom, we can do the same with $\neg B$.

2. $\underline{B \text{ is } \neg E}$, so we have:

$$\dfrac{\dfrac{\vdots}{C \vee \neg E}\,\text{-},\, 2^{\alpha_1}, m \qquad \dfrac{\vdots}{\neg\neg E \vee D}\,\text{-},\, 2^{\alpha_2}, m}{C \vee D}\,\text{Cut},\, 2^{\alpha}, m+1$$

We claim that we can make this last Cut with E as the Cut formula in place of $\neg E$, thereby bounding its degree by m instead of $m+1$.

Since $PA_\omega \vdash_m^{2^{\alpha_2}} \neg\neg E \vee D$, by Lemma 6(a) we have $PA_\omega \vdash_m^{2^{\alpha_2}} E \vee D$. Therefore we can make the derivation:

$$\dfrac{\dfrac{\dfrac{\vdots}{E \vee D}\,\text{-},\, 2^{\alpha_2}, m}{D \vee E}\,\text{Exchange},\, 2^{\alpha_2}, m \qquad \dfrac{\dfrac{\vdots}{C \vee \neg E}\,\text{-},\, 2^{\alpha_1}, m}{\neg E \vee C}\,\text{-},\, 2^{\alpha_1}, m}{D \vee C}\,\text{Cut},\, 2^{\alpha}, m}{C \vee D}\,\text{Exchange},\, 2^{\alpha}, m$$

3. $\underline{B \text{ is } E \vee F}$, so we have:

$$\dfrac{\dfrac{\vdots}{C \vee (E \vee F)}\,\text{-},\, 2^{\alpha_1}, m \qquad \dfrac{\vdots}{\neg(E \vee F) \vee D}\,\text{-},\, 2^{\alpha_2}, m}{C \vee D}\,\text{Cut},\, \alpha, m+1$$

Now, since $PA_\omega \vdash_m^{2^{\alpha_2}} \neg(E \vee F) \vee D$, by Lemma 6(b) we have

$$PA_\omega \vdash_m^{2^{\alpha_2}} \neg E \vee D, \neg F \vee D$$

Therefore we can make the derivation:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\vdots}{C \vee (E \vee F)}\,\text{-},\, 2^{\alpha_1}, m}{C \vee E \vee F}\,\text{Associativity},\, 2^{\alpha_1}, m \quad \dfrac{\vdots}{\neg F \vee D}\,\text{-},\, 2^{\alpha_2}, m}{C \vee E \vee D}\,\text{Cut},\, \max\{2^{\alpha_1}, 2^{\alpha_2}\}+1, m}{C \vee D \vee E}\,\text{Exchange},\, \max\{2^{\alpha_1}, 2^{\alpha_2}\}+1, m \quad \dfrac{\vdots}{\neg E \vee D}\,\text{-},\, 2^{\alpha_2}, m}{C \vee D \vee D}\,\text{Cut},\, \max\{2^{\alpha_1}, 2^{\alpha_2}\}+2, m}{D \vee C \vee D}\,\text{Exchange},\, \max\{2^{\alpha_1}, 2^{\alpha_2}\}+2, m}{D \vee D \vee C}\,\text{Exchange},\, \max\{2^{\alpha_1}, 2^{\alpha_2}\}+2, m}{D \vee C}\,\text{Contraction},\, \max\{2^{\alpha_1}, 2^{\alpha_2}\}+2, m}{C \vee D}\,\text{Exchange},\, \max\{2^{\alpha_1}, 2^{\alpha_2}\}+2, m$$

Thus, $PA_\omega \vdash_m^{\max\{2^{\alpha_1}, 2^{\alpha_2}\}+2}$, so we have $PA_\omega \vdash_m^{2^\alpha}$ when

$$2^\alpha \geq \max\{2^{\alpha_1}, 2^{\alpha_2}\} + 2$$

which is true if $\alpha \geq 2$. Otherwise, $\alpha = 1$, which is why we covered this as an unconditional base case above.

4. $\underline{B \text{ is } \forall x E(x)}$, so we have:

$$\dfrac{\dfrac{\vdots}{C \vee \forall x E(x)}\,\text{-},\, 2^{\alpha_1}, m \quad \dfrac{\vdots}{\neg \forall x E(x) \vee D}\,\text{-},\, 2^{\alpha_2}, m}{C \vee D}\,\text{Cut},\, \alpha, m+1$$

Now, since $PA_\omega \vdash_m^{2^{\alpha_1}} C \vee \forall x E(x)$, Lemma 6(c) says that for any number n, we have $PA_\omega \vdash_m^{2^{\alpha_1}} C \vee E(n)$ (after applying Exchange). For any closed term t_i, we can apply $\texttt{ProofSub}_{E(n),E(t_i)}$ (with substitution target $E(n)$) to this proof. Since we are only substituting terms, we are not changing the structure of any formulas in this proof, so no special cases need to be handled in any of the rules. This means that for any closed term t_i, we can build some proof:

$$\text{-}, 2^{\alpha_1}, m \; \frac{\mathscr{P}_{(i)}}{C \vee E(t_i)} \tag{*}$$

We would like to define an operation `NegUnivTransf` in analogy to the proof of Lemma 6, and we will borrow the notation from there. If $\mathscr{P}$ is our proof of $\neg\forall x E(x) \vee D$, then we would like $\texttt{NegUnivTransf}(\mathscr{P})$ to be a proof of $C \vee D$, which we will accomplish by applying $\texttt{ProofSub}_{\neg\forall x E(x),C}$ to $\mathscr{P}$ (with substitution target $\neg\forall x E(x)$), a substitution that is compatible with every rule except Quantification.

We note that since this gets rid of the final Cut in our original proof of $C \vee D$, $\texttt{NegUnivTransf}(\mathscr{P})$ will have degree m as long as `NegUnivTransf` doesn't introduce any new Cuts of degree $> m$, so our only task is to avoid this in completing our definition of `NegUnivTransf`. With that in mind, we illustrate the case of Quantification, by an extended example. If $\mathscr{P}$ is (for some subproofs $\mathscr{P}_1, \mathscr{P}_2, \mathscr{P}_3$, formulas F_1, F_2, terms t_1, t_2, and ordinals $\beta_1 < \beta_1^* \leq \beta_2 < \beta_2^* \leq 2^{\alpha_2}$):

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\mathscr{P}_1}{\neg E(t_1) \vee F_1}\,\text{-},\ \beta_1, m}{\neg \forall x E(x) \vee F_1}\,\text{Quantification},\ \beta_1^*, m}{\dfrac{\dfrac{\mathscr{P}_2}{\neg E(t_2) \vee F_2}\,\text{-},\ \beta_2, m}{\neg \forall x E(x) \vee F_2}\,\text{Quantification},\ \beta_2^*, m}}{\dfrac{\mathscr{P}_3}{\neg \forall x E(x) \vee D}\,\text{-},\ 2^{\alpha_2}, m}}$$

Then, besides applying $\texttt{ProofSub}_{\neg \forall x E(x), C}$, we also want to replace each instance of Quantification with a Cut of degree m.[4] To a first approximation, we will let $\texttt{NegUnivTransf}(\mathscr{P})$ be:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\mathscr{P}_{(1)}}{C \vee E(t_1)}\,\text{-},\ 2^{\alpha_1}, m \quad \dfrac{\mathscr{P}_1}{\neg E(t_1) \vee F_1}\,\text{-},\ \beta_1, m}{C \vee F_1}\,\text{Cut},\ \beta_1^*, m}{\dfrac{\mathscr{P}_2}{\neg E(t_2) \vee F_2}\,\text{-},\ \beta_2, m}}{C \vee F_2}\,\text{Cut},\ \beta_2^*, m \quad \dfrac{\mathscr{P}_{(2)}}{C \vee E(t_2)}\,\text{-},\ 2^{\alpha_1}, m}}{\dfrac{\mathscr{P}_3}{C \vee D}\,\text{-},\ 2^{\alpha_2}, m}$$

where we are using (*) repeatedly. The only problem here is that this is no longer necessarily a valid proof because of the ordinals: we don't necessarily have $\beta_i^* > \max\{2^{\alpha_1}, \beta_i\}$. Because this problem runs all throughout the proof, we must now change the ordinals wholesale to remedy this. Specifically, if γ is an ordinal on any part of this proof (except one of the $C \vee E(t_i)$'s on the left side), then convert γ to $2^{\alpha_1} + \gamma$. The result is:

[4] We can imagine other configurations, such as having more instances of Quantification or having them on multiple branches of a proof if they occur before another Cut, DeMorgan, or even the ω-Rule.

$$\dfrac{\dfrac{\dfrac{\dfrac{\mathscr{P}_{(1)}}{C \vee E(t_1)}\,\text{-},\, 2^{\alpha_1}, m \quad \dfrac{\mathscr{P}_1}{\neg E(t_1) \vee F_1}\,\text{-},\, 2^{\alpha_1}+\beta_1, m}{\dfrac{C \vee F_1}{\mathscr{P}_2}}\,\text{Cut},\, 2^{\alpha_1}+\beta_1^*, m \quad \dfrac{\mathscr{P}_{(2)}}{C \vee E(t_2)}\,\text{-},\, 2^{\alpha_1}, m \quad \dfrac{\mathscr{P}_2}{\neg E(t_2) \vee F_2}\,\text{-},\, 2^{\alpha_1}+\beta_2, m}{\dfrac{C \vee F_2}{\mathscr{P}_3}}\,\text{Cut},\, 2^{\alpha_1}+\beta_2^*, m}{C \vee D}\,\text{-},\, 2^{\alpha_1}+2^{\alpha_2}, m$$

We note that since $\gamma_2 < \gamma_1$ implies $2^{\alpha_1} + \gamma_2 < 2^{\alpha_1} + \gamma_1$, we have not damaged the ordinal structure we already had in $\mathscr{P}_i$, and in our new Cuts, we now have $2^{\alpha_1} + \beta_i^* > \max\{2^{\alpha_1}, 2^{\alpha_1} + \beta_i\}$ as desired.

Finalyy, our original claim was that our new proof would have ordinal $\leq 2^\alpha$, where $\alpha > \max\{\alpha_1, \alpha_2\}$. This follows from:

$$2^{\alpha_1} + 2^{\alpha_1} \leq 2^{\max\{2^{\alpha_1}, 2^{\alpha_2}\}} + 2^{\max\{2^{\alpha_1}, 2^{\alpha_2}\}} = 2^{\max\{2^{\alpha_1}, 2^{\alpha_2}\}} \cdot 2 = 2^{\max\{2^{\alpha_1}, 2^{\alpha_2}\}+1} \leq 2^\alpha$$

□

Corollary 5. *If* $PA_\omega \vdash_m^\alpha A$, *then* $PA_\omega \vdash_0^{2^{\cdot^{\cdot^{2^\alpha}}}} A$ (a tower of m 2's)

Proof. Apply the previous theorem m times. □

Corollary 6. PA_ω *(and hence* PA*) is consistent*

Proof. Reiterating our observations in Section 2.2, if PA_ω were inconsistent it would prove a dangerous disjunction A, and by the last corollary, this would mean A has a Cut-free proof. But A can't be an axiom of PA_ω, and all of the non-Cut rules can't possibly bring a derivation into danger, so this situation is impossible.

So PA_ω does not prove a contradiction, and we showed in Section 2.3 that it proves everything PA proves, so PA does not prove a contradiction either. □

CHAPTER 3

FORMALIZING GENTZEN'S PROOF IN COQ

Here we describe our implementation of Chapter 2 in the Coq theorem prover, a strongly typed functional programming language designed for expressing mathematical assertions and verifying proofs. It is among the most well-known proof assistants, having won the 2013 ACM Software System Award, and has been used to formally verify a wide and extensive range of theorems in all branches of mathematics, such as the Brouwer fixed-point theorem, Abel-Ruffini theorem, and prime number theorem.

Most famously, it was used to give the first surveyable proof of the 4-color theorem in 2005. First proved in 1979 by having a large computer program verify the thousands of combinatorial cases involved, many critics objected to the infeasibility of having a human check that the computation steps were correct, or that the special-purpose code was properly written to do what it was intended for. In contrast, when Gonthier and Werner formalized a proof in Coq, the correctness of their program depended (almost) only on the correctness of the general-purpose Coq kernel, which itself is quite amenable to human verification. As Gonthier described their result [11]:

> Even though the correctness of our proof still depends on the correct operation of several computer hardware and software components (the processor, its operating system, the Coq proof checker, and the Ocaml compiler that compiled it), none of these components are specific to the

> proof of the Four Colour Theorem. All of them come off-the-shelf, fulfill a more general purpose, and can be (and are) tested extensively on numerous other jobs, probably much more than the mind of an individual mathematician reviewing a proof manuscript could ever be. In addition, the most specific component we use–the Coq system, to which the script is tuned–can output a proof witness, i.e., a longhand detailed description of the chain of formal logical steps that was used in the proof. This witness can, in principle, be checked independently (technically, it is a term in a higher-order lambda calculus). Because this witness records only logical steps, and not computation steps, its size remains reasonable, despite the large amount of computation needed for actually checking the proof.

The Coq kernel itself is a set of axioms; specifically, the formal system known as the Calculus of inductive Constructions (CoC) developed by Thierry Coquand, both of which lend their name to Coq. CoC is a very powerful system, but it is also constructive, which means that proofs in it have a computational interpretation, most particularly[1] :

1. A proof of $P \to Q$ is a (computable) function that inputs any proof of P and outputs a proof of Q.

2. A proof of $\forall x \in S, P(x)$ is a (computable) function that inputs any $s \in S$ and outputs a proof of $P(s)$.

3. A proof of $P \vee Q$ is an ordered pair (b, p), where either $b = 0$ and p is a proof of P, or $b = 1$ and p is a proof of Q.

[1]To be precise, we are sketching out the Brouwer-Heyting-Kolmogorov (BHK) interpretation, later (and independently) developed further as the Curry-Howard correspondence in type theory.

4. A proof of $\exists x \in S, P(x)$ is an ordered pair (s, p), where $s \in S$ and p is a proof of $P(s)$.

This has a few nice consequences for Coq. From (1-2), any implication or universally quantified statement can be proved in Coq by simply writing the appropriate computable function, i.e. computer program. Moreover, since the logic is constructive, even proofs of disjunctions or existentials, as in (3-4), yield specific objects witnessing them, giving us the "proof witnesses" mentioned by Gonthier above.

This is the system in which we've approached Gentzen's consistency proof, and our code is available at:

github.com/Morgan-Sinclaire/Gentzen/blob/master/gentzen.v

As of this writing, it stands at about 5000 lines,[2] which is still a few thousand lines short of completion. However, the majority of this is routine work, and in our assessment the hardest problems have been solved. For the remainder of this chapter, we substantiate this claim, and walk through our implementation in detail. The following 11 sections correspond to the 11 sections in the code.

Though we will explain some of the more essential aspects of Coq's syntax, we cannot hope to give a comprehensive introduction in this space. For those who desire a deeper understanding, we estimate that Chapters 1-7 of Pierce's *Software Foundations* [24] is necessary and sufficient to understand virtually all the details of our implementation.

3.1 Basic Properties of Natural Numbers and Lists

The first 4 lines of the preamble read as follows:

[2] On our personal machine, this takes about 4 minutes to compile to the end.

```
Require Import Omega.

Require Import Lia.

Notation "b1 && b2" := (andb b1 b2).

Notation "b1 || b2" := (orb b1 b2).
```

Omega and Lia are two standard arithmetic libraries which we'll find convenient, particularly in this section. Crucially, they are weak systems of arithmetic contained in PRA and do not invoke the full strength of PA. Lines 3-4 are simply defining shorthand notations for Boolean operations. It is worth noting here that Booleans and propositions are treated differently in Coq, with `bool` and `Prop` being entirely different types:

	bool	**Prop**
Inhabitants	true, false	0 = 1, (forall n, n = n), etc.
Decidable?	Yes	No
Logical Connectives	negb, orb, andb, leb, eqb	$\sim, \vee, \wedge, \rightarrow, \leftrightarrow$

Thus, `bool` is just the set {true, false}, and the usual rules of classical logic apply to those values, so reasoning about it is decidable and usually trivial. However, `Prop` consists of any mathematical statement that can be expressed in Coq, and hence is not decidable.

It is now worth skipping to the proof of `eq_refl` on line 28:

```
Lemma eq_refl :  forall (n :  nat), n = n.  Proof.  auto.  Qed.
```

This is a theorem[3] which simply says that for every natural number n, $n = n$ holds. To Coq, this is actually a function which takes any $n \in \mathbb{N}$ as input and returns a proof of $n = n$. In either case, the proof is trivial, and there is a *proof tactic* (or just *tactic*, as Coq commands are called) called `auto` which can complete many trivial proofs like this.

The proof on line 32 for `addends_leq` is also easy,but not quite trivial enough for `auto`:

```
Lemma addends_leq :  forall (m n p :  nat), n + m = p -> n <= p ∧ m <= p.
Proof.  intros.  omega.  Qed.
```

Instead, we call `intros`, which effectively introduces the objects n, m, p, much like saying "let n, m, p be arbitrary". `intros` also introduces the hypothesis $n + m = p$, essentially saying "assume $n+m = p$", and leaving as our goal to prove $n \leq p \wedge m \leq p$ from this assumption. For this, we can call `omega`, the arithmetic package which can solve any quantifier-free equation that involves only addition (hence `intros` was necessary to call before this).

In `eq_nat_refl` on line 35:

[3]In Coq syntax, `Theorem`, `Lemma`, `Corollary`, and `Proposition` are synonyms.

```
Lemma eq_nat_refl :  forall (n :  nat), eq_nat n n = true.
Proof.
intros.  induction n as [| n IH].
- auto.
- simpl.  apply IH.
Qed.
```

we are proving that *boolean* equality of numbers is reflexive, rather than *propositional* equality as in `eq_refl`. Thus `eq_nat n n` is a `bool`, which is decidable, while `n = n` is a `Prop`. Since Coq requires every function written to be provably computable, it has a strong type-checking system to ensure that, and as part of this, at certain places in function definitions objects of type `Prop` are not allowed but `bool` is.

Since this is a different theorem, `eq_nat_refl` needs its own proof, and it turns out that `auto` or even `omega` can't solve this immediately. Instead we will call the `induction` tactic, as we will for almost every proof we do about the natural numbers (and many other data structures as well). The `as` keyword simply allows us to designate `n` as the variable name inside the inductive step of our proof, and `IH` as the name of our induction hypothesis. After calling `induction`, we have two cases left[4]: the base case $n = 0$ and the inductive step. The base case is trivial, while the inductive case just needs to be simplified with `simpl` before we can `apply` our induction hypothesis.

Next comes our first (non-theorem) definition:

[4]We separate these by a dash `-`. If we have nested subcases, we will then use `+`, `*`, and then simply brackets {}.

```
Fixpoint geq_nat (n m :  nat) :  bool :=
match (n, m) with
| (0, 0) => true
| (S n', 0) => true
| (0, S m') => false
| (S n', S m') => geq_nat n' m'
end.
```

Such definitions are specified in Coq using either the keywords `Definition` or `Fixpoint`, with the latter being strictly stronger in that it can do recursive definitions, which is what we're doing here in defining boolean $\geq$ in a decidable manner. The way Coq defines numbers, every n is either 0 or Sn' for some other number n', and the `match` keyword allows us to reason about these cases for both n and m. In each case, we must return a boolean, or recursively call our function in a way that Coq knows will terminate.

Our proof of `succ_geq` proceeds much as our last proof, except here we use the `rewrite` tactic

```
rewrite <- IHn.
```

which rewrites the equation we have as our goal using the equation `IHn` that we have. The `<-` indicates we have the right hand side of `IHn` in our goal, and we want to turn it into the left hand side. If `<-` is omitted, this is the other way around.

Things proceed as usual until line 74, where our first nontrivial proof begins, of:

```
Lemma lt_nat_decid :  forall (n m :  nat), n < m -> lt_nat n m = true.
```

which simply says that if the propositional $<$ relation holds between two numbers, then so does the corresponding boolean relation. To do this, we first define a number n to be `lt_nat_decid_nice` if this property holds between n and every other number n:

```
Definition lt_nat_decid_nice (n :  nat) :=
forall (m :  nat), n < m -> lt_nat n m = true.
```

Our strategy, then, is to prove that every n is `lt_nat_decid_nice`. Here we again proceed by induction, although more is going on in this proof than before, such as the snippet:

```
- unfold lt_nat_decid_nice.  intros.  destruct m.
 + inversion H.
 + unfold lt_nat_decid_nice in IHn.
   assert (n < m).  { omega.}
   apply IHn in H0.  simpl.
```

The tactic `unfold` will unwind the definition of `lt_nat_decid_nice` so we can use it. `destruct` is similar to `induction` in that it breaks m into the cases where it is 0 or Sm', except we don't get an induction hypothesis for the latter (since we don't need it in this case). `inversion`, when called on the hypothesis H, essentially unwinds the "possible reasons why" H holds. In this case H is $0 < 0$ which never holds, and so

inversion simply notes that we have a contradiction, and we are done with this case. The `assert` tactic lets us claim $n < m$, which we then quickly prove inside the curly braces. We do this whenever we feel some minor claim could help us finish a proof, and this claim is to simple and specific to be worth proving as a separate lemma. And indeed, we put this claim to good use after we've proved it as *H0*.

The next couple hundred lines proceed more or less similarly, as we prove other lemmas about numbers that we will need later (we emphasize again that none of these require machinery that isn't primitive recursive).

Beginning on line 264, we start to look at lists, primarily lists of numbers. First we define inductively what a list of X is (X will almost always be `nat` for our purposes) for this purpose the `Inductive` keyword is used to build data structures recursively:

```
Inductive list (X : Type) :  Type :=
| nil :  list X
| constr :  X -> list X -> list X.
```

In the remaining 300 lines of the section, we set our own convenient notation for lists, define some operations on lists, and prove some basic properties about them.

3.2 Ordinals up to ϵ_0

Section 2, which runs from lines 643 to 1800, involves setting up the machinery behind ordinal arithmetic that we need in Cut-elimination. Defining ordinals properly turned out to be a very difficult task. One of the difficulties was simply in getting up to $\omega \cdot 2$; for instance, if one naïvely defines the following:

```
Inductive ord :  Set :=
| Zero :  ord
| succ :  ord -> ord
| omega :  ord.
```

Then one can get every ordinal up to but not including $\omega \cdot 2$. It is worth noting that in traditional ZFC, $\omega \cdot 2$ is the first ordinal that requires the Replacement axiom to construct. The use of Replacement is essentially in saying that if we have any sequence, such as $\omega, \omega + 1, \omega + 2, ...$, then we can put those into the set

$$\{\omega, \omega + 1, \omega + 2, ...\} = \omega \cdot 2$$

and so building $\omega \cdot 2$ is easy. And in fact, in virtually every standard reference on ordinal arithmetic, it is simply assumed that we can collect sequences like this, an option not available in Replacement-free set theory, let alone the system PRA.

In our present implementation, an ordinal is defined inductively to be either `0` or `cons a n b`, which we take to represent $\omega^a \cdot (n+1) + b$, where n is a number and a, b are ordinals. In the Coq syntax, this is:

```
Inductive ord :  Set :=
| Zero :  ord
| cons :  ord -> nat -> ord -> ord.
```

For instance, any natural number m can be represented as either `Zero` or (if

$m = m' + 1$) as `cons Zero m' Zero`, i.e. $\omega^0 \cdot (m' + 1) + 0$. However, the second and even more difficult problem was defining an ordering on this new type `ord`. Under our definition, ordinals need not be in Cantor normal form, since $\omega + \omega^3$ is a valid member of `ord`. Hence, if we want to compare it with ω^2 and say the former is bigger, we cannot simply match them by the first term. We also cannot simply write a function to put $\omega + \omega^3$ in Cantor normal form, since that would imply we had a subroutine to determine that $\omega < \omega^3$, but we don't have such an ordering in the first place.

Our leading idea was to *define* ordinals to be in Cantor normal form, by defining `ord` mutually recursively with `ord_lt`. We worked on this for weeks, but it proved rather complicated, and eventually we looked to see if anyone had already attempted to do this in Coq, and it turned out that Pierre Castéran (with others) had. Castéran is a highly respected figure in the Coq community, having co-authored the authoritative text for advanced Coq users.[2] Castéran's code[5] used some very heavy-duty tactics and notations that initially left this author defeated (even after having gone through [24]). Over time, however, we came to appreciate the stunning cleverness of what was done, and copied Castéran's ordering relation into our own code:

```
Inductive ord_lt :  ord -> ord -> Prop :=
| zero_lt :  forall a n b, Zero < cons a n b
| head_lt :  forall a a' n n' b b', a < a' -> cons a n b < cons a' n' b'
| coeff_lt :  forall a n n' b b', (n < n')%nat -> cons a n b < cons a n' b'
| tail_lt :  forall a n b b', b < b' -> cons a n b < cons a n b'
where "o < o'" := (ord_lt o o') :  cantor_scope.
```

[5] Available at `www.labri.fr/perso/casteran`, under "Ordinal notations and rpo". The file that we borrow from is EPSILON0.v, in the epsilon0 folder.

In this definition, `where` indicates that in the definition, the `<` will be taken to mean `ord_lt` instead of the usual less than relation on the natural numbers. The `cantor_scope` part defines a new *scope* we can open up if we want to, and in this scope, $<$ will have that meaning. Indeed, immediately after this definition we call `Open Scope cantor_scope.`, and we close this at the very end of Section 2. In addition, `(n < n')%nat` is syntax to override the `where` statement, so that the $<$ symbol in parentheses *will* be taken to mean the usual less than relation on `nat`.

Turning back to the core definition itself, one will notice that it does not actually correspond to the definition we actually want. In particular, if we have:

$$\alpha = \omega^0 \cdot (3+1) + \omega^0 \cdot (2+1) = \texttt{cons Zero 3 (cons Zero 2 Zero)}$$
$$\beta = \omega^0 \cdot (4+1) = \texttt{cons Zero 4 Zero}$$

Then $\alpha < \beta$ holds according to `ord_lt` because of its `coeff_lt` constructor since $3 < 4$ as numbers. But this is equivalent to saying that, as ordinals, $4 + 3 < 5$.

In Castéran's file, this odd definition does not endear itself until a few hundred lines later, when Cantor normal form is defined.[6] This definition, which we we also copied, runs as follows on line 813:

[6]In our implementation, we have also found it more natural to first spend a few hundred lines proving useful order-theoretic properties about `ord_lt`, such as transitivity, irreflexivity, and completeness (connexivity).

```
Inductive nf :  ord -> Prop :=
| zero_nf :  nf Zero
| single_nf :  forall a n, nf a -> nf (cons a n Zero)
| cons_nf :  forall a n a' n' b,
    a' < a -> nf a -> nf (cons a' n' b) -> nf (cons a n (cons a' n' b)).
```

And we can now notice that in our previous example with $\alpha < \beta$, α is not in normal form, and in general, the `ord_lt` relation *does* correspond to the usual $<$ relation on ordinals if they are in normal form. At this point, if we desired, we could define a new ordinal $<$ relation that works for all ordinals now that we have a proper definition of normal form (and go on to prove that this new definition matches our expectations), but for the sake of Cut-elimination we only need to consider ordinals in normal form.

The point here is that by "cheating" with an only mostly-accurate `ord_lt` relation, Castéran was then able to define a fully accurate `nf` definition. This is because the latter only invokes the former once, in the $a' < a$ condition of `cons_nf`, where `ord_lt` actually is accurate. In this way, mutual recursion is avoided, which we had personally found to be a rather awkward construction to attempt.

Our next few hundred lines involve defining boolean equality and less than relations on ordinals. Like many of our other definitions in this section, these assume normal form, and will not necessarily work as intended otherwise. We also prove further order-theoretic properties of ordinals, and similarly many of these theorems don't have the normal form assumption, even if we only had normal form ordinals in mind as we stated and proved them.

On lines 1158-1198 we actually define the ordinal arithmetic operations of addition, multiplication, and exponentiation (assuming normal form). In most standard references (e.g. [17]) they are defined as follows, for ordinals α, β, γ:

Definition (Ordinal Addition)**.**

$$\begin{aligned} &(i) && \alpha + 0 = \alpha \\ &(ii) && \alpha + (\beta + 1) = (\alpha + \beta) + 1 \\ &(iii) && \alpha + \beta = \sup\{\alpha + \gamma \mid \gamma < \beta\} \end{aligned}$$

Definition (Ordinal Multiplication)**.**

$$\begin{aligned} &(i) && \alpha \cdot 0 = 0 \\ &(ii) && \alpha \cdot (\beta \cdot 1) = \alpha \cdot \beta + \alpha \\ &(iii) && \alpha \cdot \beta = \sup\{\alpha \cdot \gamma \mid \gamma < \beta\} \end{aligned}$$

Definition (Ordinal Exponentiation)**.**

$$\begin{aligned} &(i) && \alpha^0 = 1 \\ &(ii) && \alpha^{\beta+1} = \alpha^\beta \cdot \alpha \\ &(iii) && \alpha^\beta = \sup\{\alpha^\gamma \mid \gamma < \beta\} \end{aligned}$$

In each of these, the computational difficulty lies with taking supremums, because it has to be rigorously shown that 1) there *is* and upper bound on any such sequence, and 2) there is a *least* one. While (2) follows from the well-foundedness of ordinals,[7]

[7]Or, in our case, our axiom of well-foundedness below ϵ_0

(1) depends on Replacement as we noted earlier.

Thus we had to take a different approach, where the arithmetic operations are provably computable. For addition, we ultimately arrived at:

```
Fixpoint ord_add (alpha beta :  ord) :  ord :=
match alpha, beta with
| _, Zero => alpha
| Zero, _ => beta
| cons a n b, cons a' n' b' =>
    (match (ord_ltb a a') with
    | true => beta
    | false =>
        (match (ord_eqb a a') with
        | true => cons a' (n + n' + 1) b'
        | false => cons a n (ord_add b beta)
        end)
    end)
end.
```

Syntactically, this is simply a nested `match` statement that covers every case for `alpha, beta`, with the underscores `_` denoting "otherwise". `ord_ltb` and `ord_eqb` are the boolean less than and equality relations we defined earlier. Our definition for multiplication is:

```
Fixpoint ord_mult (alpha beta :  ord) :  ord :=
match alpha, beta with
| _, Zero => Zero
| Zero, _ => Zero
| cons a n b, cons Zero n' b' => cons a ((S n) * (S n') - 1) b
| cons a n b, cons a' n' b' => cons (ord_add a a') n' (ord_mult alpha b')
end.
```

Both of these took some careful thought to devise, as well as to confirm that they match the standard definition. Exponentiation proved to be even harder, and after being puzzled for nearly a week, we looked to see that Casteran had in fact given a complicated but very workable definition. After copying that and specializing it to when the base is 2 (the only case we need in our proof) we got:

```
Fixpoint ord_2_exp (alpha :  ord) :  ord :=
match alpha with
| Zero => cons Zero 0 Zero
| cons Zero n' _ => nat_ord (2 ^ (S n'))
| cons (cons Zero n Zero) 0 Zero =>
    cons (cons (cons Zero n Zero) 0 Zero) 0 Zero
| cons (cons a n b) n' b' =>
    ord_mult (cons (cons (cons a n b) n' Zero) 0 Zero) (ord_2_exp b')
end.
```

The rest of this section of our implementation is devoted to showing that normal form ordinals are closed under these arithmetical operations. In the next 200 lines, we prove some miscellaneous lemmas that will help with this task, and then `nf_add`:

```
Lemma nf_add :  forall (alpha beta :  ord),
    nf alpha -> nf beta -> nf (ord_add alpha beta).
```

Our proof strategy here is similar to our `lt_nat_decid` example in the previous section: we first define α in normal form to be `nf_add_nice` if for any β in normal form, $\alpha + \beta$ is in normal form. Then, we prove by induction over `ord` that every α is `nf_add_nice`. Almost all the remaining major proofs in this section will follow this template.

Indeed, before proving the corresponding `nf_mult`, we will need to prove the following lemmas:

```
Lemma add_right_incr :  forall (alpha beta gamma :  ord),
    beta < gamma -> ord_add alpha beta < ord_add alpha gamma.
Lemma mult_right_incr :  forall (alpha beta gamma :  ord),
    beta < gamma -> Zero < alpha -> nf gamma ->
    ord_mult alpha beta < ord_mult alpha gamma.
```

But to prove the latter, we will actually need *two* auxiliary definitions:

```
Definition mult_right_nice (alpha :  ord) :=
    alpha = Zero ∨ forall (beta gamma :  ord),
    beta < gamma -> nf gamma -> ord_mult alpha beta < ord_mult alpha gamma.
Definition mult_right_nice2 (beta alpha :  ord) :=
    alpha = Zero ∨ forall (gamma :  ord),
    beta < gamma -> nf gamma -> ord_mult alpha beta < ord_mult alpha gamma.
```

With these lemmas in hand, we then proceed to prove `nf_mult` and `nf_2_exp`. Finally, we prove:

```
Lemma ord_2_exp_fp :  forall (alpha :  ord), nf alpha ->
    alpha < ord_2_exp alpha ∨ alpha = cons (nat_ord 1) 0 Zero.
```

As noted in Chapter 1, ϵ_0 is defined to be the least ordinal α that satisfies $\omega^\alpha = \alpha$, i.e. the least fixed point of the mapping $\alpha \mapsto \omega^\alpha$. It follows that it is also a fixed point of $\alpha \mapsto 2^\alpha$, but so is $\omega = 2^\omega$. The above theorem states that ω is the only ordinal $< \epsilon_0$ with this property. This is rather difficult to prove on paper with the usual ordinal exponentiation definition, since the ordinal computations quickly become unwieldy. However, with our (Castéran's) more computational definition, this becomes quite doable.

We thought this theorem would be needed since, in the Cut-elimination argument, we repeatedly apply the $\alpha \mapsto 2^\alpha$ mapping, and want to use the fact that ϵ_0 is the supremum of $\alpha, 2^\alpha, 2^{2^\alpha}, ...$ for any $\alpha > \omega$. It would follow that induction up to ϵ_0 is precisely what we need to reason about such exponentiated sequences. However, as

we came to understand the induction argument in more detail, it turned out that we did not actually need this theorem.

3.3 FOL Machinery

Now we finally talk about first-order logic, and the language of PA/PA_ω. Our first task is to inductively define terms, atomic formulas, and formulas:

```
Inductive term :  Type :=
| zero :  term
| succ :  term -> term
| plus :  term -> term -> term
| times :  term -> term -> term
| var :  nat -> term.
Inductive atomic_formula :  Type :=
| equ :  term -> term -> atomic_formula.
Inductive formula :  Type :=
| atom :  atomic_formula -> formula
| neg :  formula -> formula
| lor :  formula -> formula -> formula
| univ :  nat -> formula -> formula.
```

The only interesting feature here is how we treat variables with `var` and `univ`. We imagine that we have variable names $x_0, x_1, x_2, ...$ and so that `var n` denotes x_n,

and for any formula A, `univ n A` denotes $\forall x_n A$.

Next we inductively define `num_conn` to simply count the number of connectives/quantifiers in any formula, which we will need when we later prove theorems by induction on this value. Then we define syntactic equality of formulas inductively, by defining `eq_term`, `eq_atom`, and `eq_f`. Many of the definitions in this section will follow this pattern of induction over terms, atomic formulas, and formulas, and consequently, many of the theorems will also have to induct over all three data structures.

Next we define our `eval` function. Our intent here is to take a term and compute what number it actually is. For instance, the terms:

$$t_1 := \texttt{plus (succ zero) (succ zero)}$$

$$t_2 := \texttt{succ (succ zero)}$$

should both evaluate as 2, even though they are syntactically distinct (i.e. `eq_term` would return `false`). Since any closed term t is simply consists of sums and products of (successors of) `zero`'s, t will evaluate to some number n. On the other hand, a non-closed term such as `var n` (i.e. x_n) should not have any definite numerical value.

In defining `eval`, one option would be to send non-closed terms to some value such as `NaN`. But in that case, the range of `eval` would no longer be `nat`, but instead some *ad hoc* type consisting of the natural numbers together with this extra inhabitant `NaN`. We instead chose to preserve `nat` as the range by letting `eval(t)` be 0 if t is not closed, otherwise $1 + n$ if n is the number t evaluates to. For instance, `eval(zero)` is 1. We also define a kind of inverse function `represent` which takes a number n and returns the term that represents it, e.g. `represent(2)` is `succ (succ zero)`.

Next, we begin to build up the notion of an atomic formula being *correct* in the

sense of Section 1 of Chapter 2, so that we may ultimately define the axioms of PA_ω. Our `correctness` function calls `eval` in the appropriate manner to determine if a given atomic formula is `correct`, `incorrect`, or `undefined`.

Over the next couple hundred lines, we define the (self-explanatory) boolean predicate `closed`, as well as the function `free_list` which returns the list of (indexes of) free variables in a formula, e.g `free_list` applied to $\forall x_2 : x_7 = x_1$ would return the list of numbers $[7, 1]$. After proving some basic facts about both of these functions, we show that any formula A, `closed(a) = true` if and only if `free_list(A) = []`. With this robust notion of closedness in hand, over the next 100 lines we prove that any closed formula is either `correct` or `incorrect`:

```
Lemma correctness_decid :  forall (a :  atomic_formula),
    closed_a a = true ->
    correct_a a = true ∨ incorrect_a a = true.
```

Recall that we used this fact multiple times in our Chapter 2 proof, when we asserted that if A is closed and atomic, then either A or $\neg A$ is an axiom of PA_ω. Also recall that this is a nontrivial fact, since Coq is constructive, and so what we have shown in classical terms is that it is fully decidable whether or not a given formula is an axiom.

The remaining 100 lines of this section are concerned with defining term substitution on formulas. Following the usual template in this section, we first define `substitution_t` as term substitution for terms, then `substitution_a` as term substitution for atomic formulas, and finally:

```
Fixpoint substitution (A : formula) (n :  nat) (t :  term) :  formula :=
match A with
| atom a => atom (substitution_a a n t)
| neg B => neg (substitution B n t)
| lor B C => lor (substitution B n t) (substitution C n t)
| univ m B =>
    (match (eq_nat m n) with
    | true => A
    | false => univ m (substitution B n t)
    end)
end.
```

Thus `substitution A n t` means $A[x_n/t]$ "start with formula A, and replace every free occurrence of x_n with the term t." To conclude the section, we show that if t is a closed term and A is a formula with x_n as its only free variable, then $A[x_n/t]$ is closed:

```
Lemma one_var_free_lemma :  forall (A : formula) (n :  nat) (t :  term),
    closed_t t = true ->
    free_list A = [n] ->
    closed (substitution A n t) = true.
```

3.4 The System PA_ω

In this section we build PA_ω as a deductive system. From our work in the previous section, its now easy enough to tell if a given formula A is an axiom:

```
Definition PA_omega_axiom (A : formula) :  bool :=
    match A with
    | atom a => correct_a a
    | neg (atom a) => incorrect_a a
    | _ => false
end.
```

And theorems of PA_ω are defined inductively: A is a theorem if its an axiom, or if A is the result of applying a rule of inference to some theorem A'. Thus our definition begins:

```
Inductive PA_omega_theorem :  formula -> Prop :=
| axiom :  forall (A : formula),
    PA_omega_axiom A = true ->
    PA_omega_theorem A
| exchange1 :  forall (A B : formula),
    PA_omega_theorem (lor A B) ->
    PA_omega_theorem (lor B A)
| exchange2 :  forall (C A B : formula),
    PA_omega_theorem (lor (lor C A) B) ->
    PA_omega_theorem (lor (lor C B) A)
```

There are a few things to notice here. First of all, for a formula A, `PA_omega_theorem` is of type `Prop`, while `PA_omega_axiom` was of type `bool`. This is because it is undecidable to tell if a given formula A is a theorem (this is called the *Entscheidungsproblem*). After all, if we could computably decide (from within PRA) whether or not $0 = 1$ is a theorem, then we would not have to go to all this trouble proving this system's consistency in the first place. So, the best we can do is show on a case by case basis that `PA_omega_theorem(A)` holds for a given A, but we won't be able to show `PA_omega_theorem(A)` is false without proving the consistency of PA_ω in the first place.

The other important observation is that `exchange1` and `exchange2` are both instances of the exchange rule. In fact, our full definition has 4 different instances, and these correspond to all the possibilities we get from C, D being present or absent in the way we defined the rule at the beginning of Chapter 2. We were initially tempted to

keep this to one instance by saying that Exchange always takes arguments C, A, B, D, and sometimes C, D are $0 = 0$. However, this scheme is not equivalent, because we actually don't have a way of eliminating these $0 = 0$'s without using the Cut rule, and this causes Cut-elimination to fail.

Hence, we are stuck with having multiple non-redundant instances of many of the other rules too, for when the side formulas are present or absent. In total, this gave us 18 rules, and hence (including `axiom`) 19 different constructors for `PA_omega_theorem`. This means that every time we define or prove something inductively over `PA_omega_theorem`–as we often will from this point on–we have 19 different cases we have to address. Albeit many of these cases are similar, so copying and pasting code across some of these cases often goes a long way.

For the most part, defining the rest of the inference rules was straightforward. It is worth remarking that quantification required our machinery for `closed` and `substitution`:

```
| quantification1 :  forall (A : formula) (n :  nat) (t :  term),
    closed_t t = true ->
    PA_omega_theorem (neg (substitution A n t)) ->
    PA_omega_theorem (neg (univ n A))
```

But certainly the most nontrivial rule to implement is the ω-Rule:

$$\frac{A(m) \text{ for each } m \in \mathbb{N}}{\forall m A(m)}$$

Recall that we can think of the ω-Rule as taking in a function g, where for any

m, $g(m)$ is a proof of $A(m)$. More precisely though, g exists in our metatheory, and so really $g(m)$ should return a proof of $PA_\omega \vdash A(m)$. But how do we tell if a given g is indeed such a function?

First of all, we need g to be total; a natural class of total functions is the primitive recursive functions. These have the additional property that they are exactly the functions that PRA *proves* total. On the other hand, our metatheory here is actually Coq, and the Calculus of Constructions is a much stronger system than PRA, with the ability to prove many non-primitive recursive functions total as well. Although we won't need any of these, and every use of the ω-Rule in chapter 2 only used a primitive recursive g, it will be convenient for us to simply use Coq's native syntax rather than defining the primitive recursive functions ourselves.

The other point about g is that we need all of its outputs to be proofs, and in particular $g(m)$ must always be a proof of $PA_\omega \vdash A(m)$. That is to say, $g(m)$ must be the correct `Type`. But this is where Coq shines, as does the BHK interpretation we mentioned at the beginning of this chapter, when we said:

> A proof of $\forall x \in S, P(x)$ is a (computable) function that inputs any $s \in S$ and outputs a proof of $P(s)$.

Taking $S = \mathbb{N}$ and $P(m)$ as $PA_\omega \vdash A(m)$, we have:

> A proof of $\forall m \in \mathbb{N}, PA_\omega \vdash A(m)$ is a (computable) function that inputs any $m \in \mathbb{N}$ and outputs a proof of $PA_\omega \vdash A(m)$.

In other words if we want our computable function g, all we have to do is prove the theorem $\forall m \in \mathbb{N}, PA_\omega \vdash A(m)$. So if we managed to prove (for the appropriate variable x_n):

```
Theorem g :  forall (m :  nat),
    PA_omega_theorem (substitution A n (represent m)).
```

Then we have an object of type $\forall m \in \mathbb{N}, PA_\omega \vdash A(m)$, namely g, which is regarded both as a proof of this universal statement, *and* as a function that inputs m and outputs $PA_\omega \vdash A(m)$. To Coq, these are synonymous, and so type theory makes our definition of the ω-Rule simple and elegant:

```
| w_rule1 :  forall (A : formula) (n :  nat)
   (g :  forall (m :  nat),
       PA_omega_theorem (substitution A n (represent m))),
   PA_omega_theorem (univ n A)
```

Alas, there is one final and altogether different complication with the ω-Rule. We noted as an afterthought in section 4 of chapter 2 that we need proofs in the ω-Rule to have a uniform bound M on the degree of Cuts used in each PA_ω proof of $A(0), A(1), ...$ Now, the theory-metatheory distinction isn't an obstacle: since we're working in constructive logic, g will produce witnesses, i.e. we can extract actual PA_ω proof objects and inspect their degree. The only issue here is we haven't included degree in our definition here of `PA_omega_theorem`.

For that matter we haven't included ordinals either, and for the same reason: these proof objects are already relatively cumbersome to reason about in Coq, with the 19 cases we have to deal with in each definition and proof. Our approach has been to develop PA_ω without these tedious ornaments first, so that we can tackle the harder problems with less awkwardness, and then add these onto our machinery later,

making the requisite minor changes to our proofs/definitions at that time. We said at the outset that this work is unfinished, and so while we have basically accomplished the former, we have yet to do the latter, and the rest of the present implementation that we discuss in this chapter will not have degrees or ordinals attached to proofs.

After a brief 50-line example illustrating the use of the ω-Rule, we get to our first actual proof of one of our results from chapter 2. Namely, Proposition 2 from 2.3, that PA_ω proves the associative rules, and the resemblence between our two proofs is evident:

```
Lemma associativity1 :  forall (c a b :  formula),
    PA_omega_theorem (lor (lor c a) b) ->
    PA_omega_theorem (lor c (lor a b)).
Proof.
intros.
apply exchange3 in H.
apply exchange2 in H.
apply exchange1 in H.
apply H.
Qed.
```

That is, after introducing our hypothesis `H`, we simply apply 3 different instances of the exchange rule in sequence to it, until `H` is identical with our goal so we can apply it (the proof of the other associative rule is similar).

The remaining 300 lines in the section merely proving some miscellaneous lemmas

we will need in the next section. Most of these are very simple facts about term substitution that are hardly ever explicitly mentioned in standard logic texts, let alone proved. For instance,

```
Lemma closed_subst_eq :  forall (A : formula) (n :  nat) (t :  term),
    closed A = true -> substitution A n t = A.
```

Took 76 lines if we include the auxiliary lemmas needed for it, which in turn had to be proved by induction over terms, atomic formulas, and formulas. There may well be implementations of FOL where this lemma really is trivial, but in our case we often had to reason often about our free variable lists, and in this regard put to good use the elementary list lemmas we proved in section 1.

3.5 PA_ω proves LEM

The aim of this section is to prove Lemma 1 from Chapter 2:

Lemma 8. *For any closed terms s and t, if $s = t$ is correct and $A(x)$ is a formula with x the only free variable (or A is closed), then $PA_\omega \vdash \neg A(s) \vee A(t)$.*

Back there, we proved this, and then as a corollary proved that $PA_\omega \vdash \neg A \vee A$, and we will call these results LEM_term and LEM, respectively. In our implementation, we opted to prove LEM first, and then LEM_term completely separately. The first reason for this is that it was not (and is not) clear to us if LEM would follow as an easy corollary the way it did on paper; as discussed above with `closed_subst_eq`, reasoning formally about substitution is rather complicated, even for closed formulas. But the more important reason for this order is that we simply tried proving LEM

first to see if it would work–as getting this right was already nontrivial–and then we could use these proof strategies for LEM_term, copying over the relevant code so and then handling the substitution differences.

The inductive argument for LEM is rather subtle, at least when formalized. To a first approximation, we are inducting over the number of connectives/quantifiers in a formula. We break this up into the predicates P_1, P_2, P_3 in the code: $P_1(A)$ simply states that if A is closed, then $\neg A \vee A$ is provable, so proving $\forall A, P_1(A)$ is our ultimate goal. $P_2(A, n)$ states that for a given n, if A has that many connectives then $P_1(A)$ holds. $P_3(n)$ simply says that $P_2(A, n)$ holds for every A:

$$\begin{aligned} P_1(A): &\quad \text{closed}(A) \to (PA_\omega \vdash \neg A \vee A) \\ P_2(A, n): &\quad \text{num\_conn}(A) = n \to P_1(A) \\ P_3(n): &\quad \forall A, P_2(A, n) \end{aligned}$$

If we can prove $\forall n, P_3(n)$ then we are done, since this implies $\forall A \forall n, P_2(A, n)$ which in turn implies $\forall A, P_1(A)$ (and further down, we prove these implications quite easily in Coq as `P1_lemma`, `P2_lemma`, and `P3_lemma`). And we actually need to prove $\forall n, P_3(n)$ by strong induction, since as a careful reading of our proof of Lemma 1 in the last chapter would reveal, in order to show that $P_2(A, n+1)$ holds for an arbitrary formula A, we will need the fact that $P_1(B)$ holds for *every* formula B with $\leq n$ connectives. Now, Coq has regular induction built deeply into its syntax, but we need to explicitly prove that this implies strong induction:[8]

[8]Most of our code on this is adapted from `pldev.blogspot.com/2012/02/proving-strong-induction-principle-for.html`

```
Lemma P3_strongind_aux :
    P3 0 ->
    (forall n,
        ((forall m, m <= n -> P3 m) -> P3 (S n))) ->
    (forall n, m <= n -> P3 m).
Lemma P3_strongind :
    P3 0 ->
    (forall n,
        ((forall m, m <= n -> P3 m) -> P3 (S n))) ->
    (forall n, P3 n).
```

Which in standard notation is:

$$P_3(0) \to \forall n((\forall m \leq n, P_3(m)) \to P_3(n+1)) \to \forall n \forall m \leq n, P_3(m)$$

$$P_3(0) \to \forall n((\forall m \leq n, P_3(m)) \to P_3(n+1)) \to \forall n, P_3(n)$$

With that in mind, we finally do the main inductive step.

```
Lemma P3_inductive :  forall n, (forall m, m <= n -> P3 m) -> P3 (S n).
```

And this proof follows quite closely with our argument in chapter 2. Our immediate corollary is:

```
Lemma LEM : forall (A : formula),
    closed A = true -> PA_omega_theorem (lor (neg A) A).
Proof.  apply P1_lemma.  Qed.
```

We now move on to LEM_term, the statement that $PA_\omega \vdash \neg A(s) \vee A(t)$ when s, t are terms that evaluate to the same number. Here, the atomic case is less straightforward and takes 123 lines, most of which involves lemmas about `substitution_a`, `substitution_t`, `eval`, and `correct_a`, but our upshot is:

```
Lemma LEM_term_atomic :
    forall (a :  atomic_formula) (n :  nat) (s t :  term),
        correct_a (equ s t) = true ->
        free_list_a a = [n] ->
        PA_omega_theorem (lor (neg (atom (substitution_a a n s)))
                              (atom (substitution_a a n t))).
```

After this, the inductive argument is almost identical to that of LEM: we define `Q_1`, `Q_2`, `Q_3`, and prove strong induction on `Q_3`. The inductive step itself (`Q3_inductive`) also closely follows the argument in chapter 2, except there are a few more lines for managing substitution trivia. Ending the section, we have:

```
Lemma LEM_term :  forall (A : formula) (n :  nat) (s t :  term),
    correct_a (equ s t) = true ->
    free_list A = [n] ->
    PA_omega_theorem (lor (neg (substitution A n s)) (substitution A n t)).
Proof.  apply Q1_lemma.  Qed.
```

3.6 The System PA

This section of our code is blank as of this writing, because we have not done this yet. However, we are quite confident this will be easy: it has a similar structure to the system PA_ω, except without the ω-Rule, which we've noted was the trickiest part of the latter. One other notable difference is that PA has more axioms and only two rules of inference, but since each of the axioms have a simple formulaic structure (unlike `PA_omega_axiom`) we expect them to be even easier.

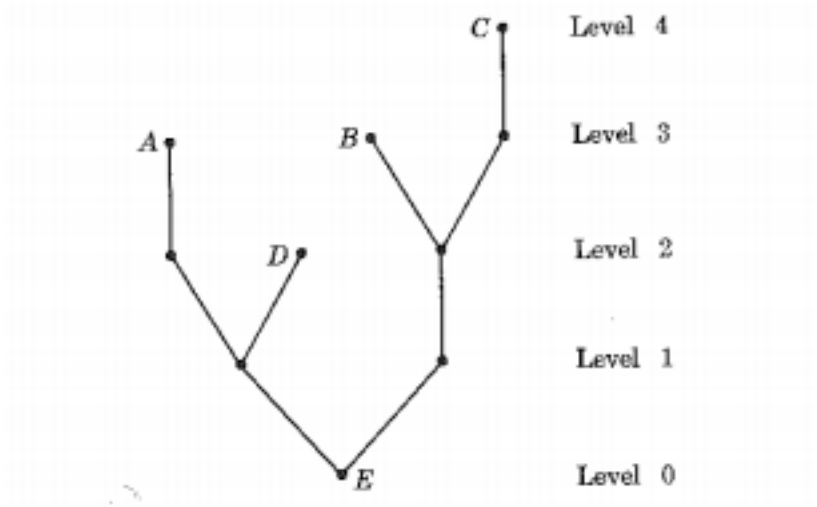
3.7 $PA \subseteq PA_\omega$

We also haven't started on this section, and of course can't until we implement PA. After finishing `LEM` and `LEM_term`, we expected (and expect) this to be straightforward, but potentially time-consuming. The reader can verify this for themselves: our proofs of Lemmas 2 and 3 simply involved building explicit proofs, which we can easily do in the same manner we implemented Proposition 2. Lemma 4 will likely take some thought and a number of auxiliary lemmas, but we would be surprised if it were more difficult than Lemma 1, given the machinery we have in place. Thus, we determined

that Cut-elimination is the more interesting and conceptually difficult part of the proof, so we undertook to first build the requisite machinery for that instead.

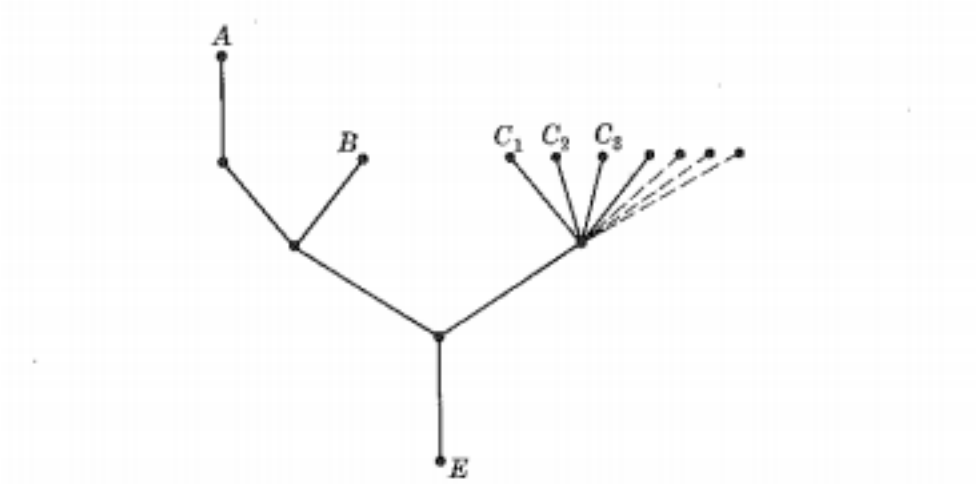
3.8 Proof Trees in PA_ω

In Mendelson's [20] formulation of Gentzen's proof, on which our own chapter 2 exposition is based, the notion of a *proof tree* is introduced first. In most first-order systems, a proof is simply viewed as a sequence of formulas with a certain structure, i.e. each is either an axiom or the result of an inference rule applied to a previous formula in the sequence. In PA_ω however, the DeMorgan and Cut rules require two premises A and B, each of which require their own proofs. This is best visualized by imagining the proof "branch out" into the two separate proofs for A and B, which are then connected by DeMorgan/Cut. In Mendelson's scheme, this is seen as:



A, B, C, D, are initial points. E is the terminal point.

Where E is the conclusion of the proof, A, B, C, D are axioms, and the 2-forked branchings are easily seen. On other hand, the ω-Rule gives us ω-forked branchings as follows:



A, B, C_1, C_2, C_3,...are the initial points. E is the terminal point.

Where $C_1, C_2, C_3, ...$ are premises in the ω-Rule which themselves have their own proofs. These proofs, in turn, may use the ω-Rule, so our tree might have several ω-forks in a row. In addition, Mendelson also defines the degree and ordinal of a proof tree right from the beginning. This of course differs from our own chapter 2 presentation, where we never particularly felt the need to mention proof trees by name, and we didn't discuss degrees and ordinals until they were actually needed for Cut-elimination.

Nevertheless, when we began implementing this proof, proof trees were among the first problems we tackled, and they proved to be the most difficult. To define the tree structure itself was mostly straightforward,[9] but it was harder to define it such that every member of the type had the required structure vis-à-vis the inference rules, degrees, and ordinals. We tried a simpler example, where we imagined having the type:

[9]Handling the ω-forkings did take some thought.

```
Inductive nat_btree :  Type :=
| leaf :  nat -> nat_btree
| unary :  nat_btree -> nat -> nat_btree
| binary :  nat_btree -> nat_btree -> nat -> nat_btree.
```

And wanting to define a `special_nat_btree` to be a `nat_btree` where every node is either:

1. A leaf
2. A node with value $n + 1$ and exactly one child of value n
3. A node with value n^2 and exactly two children of value n

Of course, we knew we could always define a boolean `valid` predicate on `nat_btree` to determine if a given tree matched (1-3), but it was our understanding at the time that the "professional" way to achieve our aims was by defining our *subtype* `special_nat_btree`, in which every inhabitant provably has the desired properties. We asked this on StackOverflow [31] and were informed that a certain type-theoretic structure called a Σ *type* does what we were asking for.

We then spent some time learning about Σ types, but they proved to be quite unwieldy for our purposes, due to the way that at each step, they demanded a proof that our tree was of the right structure, and we saw no convenient way to carry these around. Since one comment on our post asked why we could not accomplish our goals by simply defining the `valid` predicate, we concluded that it was not considered amateurish to do that in the situation we were in.

Even so, the way we defined proof trees, with all the inference rules as well as the degrees and ordinals, it was very difficult to do anything nontrivial with these structures because of how cumbersome they were. We more or less gave up on them for a few months, and in the intervening time studied Mendelson's proof in greater detail, built up more of the basic FOL machinery, and, most of all, struggled through ordinal definitions to produce what is now section 2.

After this extended break, we decided to drop degrees and ordinals, both as a temporary simplifying measure, but also because we noticed that the entire proof of $PA \subseteq PA_\omega$ doesn't involve these, and that none of these results are used in Cut-elimination, so we could safely do the first half of the whole proof without them anyways.

We also gave up proof trees, and instead opted to simply define theorems in PA_ω in the usual inductive manner, and this resulted in the work described in section 4. In particular, after proving the associative rules quite easily–even these trivial facts were a struggle with our original proof trees–we gained the confidence to continue further, and this resulted in proving `LEM_term`. While this was a considerable amount of work, as we described in section 5, it felt quite smooth compared to what we had been working with before.

As such, we hoped to make this second half of the proof work without having to define proof trees. But despite the considerable effort expended into this, the fact is that Lemma 6 from chapter 2 (and everything onward) absolutely requires the ability to reason about proofs as objects. Whereas the arguments behind PA_ω all involve building explicit step-by-step inferences and otherwise doing "local" actions, the invertibility lemmas involve making "global" substitutions across an entire proof in a delicate way. For this purpose, we cannot simply induct over formulas/theorems

as we had been doing previously, but we had to induct over proofs themselves.

When we built proof trees this time, now that we had several months of experience with both Coq and this proof, we were able to make things more streamlined than before. For one thing, we decided to do so without degrees/ordinals to convince ourselves that the fundamentals work, so we named our new type `ftree` ("formula tree") to emphasize that these are not real proof trees since they just have the bare, undecorated formulas. Also, we found that the most natural way to build the inference rules is to let them take as arguments one or more of the formulas A, B, C, D. For instance, `ftree` begins:

```
Inductive ftree :  Type :=
| node :  formula -> ftree
| exchange_ab :  formula -> formula -> ftree -> ftree
| exchange_cab :  formula -> formula -> formula -> ftree -> ftree
```

So that `node` simply takes a single formula (presumably an axiom), `exchange_ab` takes in the two formulas A, B that it will swap, `exchange_cab` takes in three formulas C, A, B (in that order), and similarly with the other rules. The exceptions are Quantification and the ω-Rule, which are structurally different from the other rules and so will be the exceptions in most definitions/proofs in this section. Thus further down our definition of `ftree` we find:

```
| quantification_ad :  formula -> formula -> nat -> term -> ftree -> ftree
| w_rule_a :  formula -> nat -> (nat -> ftree) -> ftree
```

`quantification_ad` inputs (in order) the principal formula A, the side formula D, the (index of the) substitution variable x_n, and the `ftree` that yielded $\neg A(t) \vee D$, giving:

$$\frac{\neg A(t) \vee D}{\neg \forall A(n) \vee D} \text{ Quantification}$$

Similarly, `w_rule_a` takes in the principal formula A, the (index of the) substitution variable x_n, and a computable function g such that $g(n)$ is a proof of $A(n)$.

The function `ftree_formula` simply returns the formula at the bottom of an `ftree`, i.e. the conclusion of a given proof, and begins:

```
Fixpoint ftree_formula (P : ftree) :  formula :=
match P with
| node A => A
| exchange_ab A B P' => lor B A
```

Next we define the predicate `valid`, which determines if a given `ftree` actually represents a proof in PA_ω, and begins:

```
Fixpoint valid (P : ftree) :  Prop :=
match P with
| node A => PA_omega_axiom A = true
| exchange_ab A B P' => ftree_formula P' = lor A B ∧ valid P'
```

Of note, Weakening requires that the new formula introduced is closed:

```
| weakening_ad A D P' => ftree_formula P' = D ∧ closed A = true ∧ valid P'
```

The ω-Rule is:

```
| w_rule_a A n g => forall (m :  nat),
    ftree_formula (g m) = substitution A n (represent m) ∧ valid (g m)
```

and we remark again that a proper definition here would mandate a uniform bound on the degrees of every $g(m)$, but we have not done this yet because we have not attached degrees to our proofs yet.

It is now natural to define:

```
Definition provable (A : formula) :  Prop :=
    exists (t :  ftree), ftree_formula t = A ∧ valid t.
Lemma provable_theorem :  forall (A : formula),
    PA_omega_theorem A -> provable A.
Proof.  Admitted.
Lemma theorem_provable :  forall (A : formula),
    provable A -> PA_omega_theorem A.
Proof.  Admitted.
```

That is, a formula A is `provable` if it is the conclusion of some `ftree`. It can then be shown that `provable(A)` holds if and only if `PA_omega_theorem(A)` holds, so that our earlier easier-to-work-with formulation of PA_ω is identical to the present one, and

so we do not have to go back and prove (e.g.) `LEM_term` with proof trees. We have not actually shown this equivalence yet, but we estimate it will be a straightforward induction over each of these 19-case definitions. The `Admitted` keyword allows us to save an incomplete proof while still using the theorem itself in future proofs. This is the first use of `Admitted` (or an equivalent) in the file, and the only other two are even more trivial.

As a proof of concept, we prove associativity with proof trees:

```
Lemma associativity_1 :  forall (C A B : formula),
    provable (lor (lor C A) B) -> provable (lor C (lor A B)).
Proof.
unfold provable.  intros.  destruct H as [t H].
eapply ex_intro.
instantiate (1:= exchange_ab (lor A B) C
                    (exchange_cab A C B
                        (exchange_abd C A B t))).
simpl.  auto.
Qed.
```

And we accomplish something we never managed do with our proof trees from months before. Still, this is less straightforward than our proof of `associativity1` back in section 4. After the `intros`, our proof window looks like the following:

```
H : exists t :  ftree, ftree_formula t = lor (lor C A) B ∧ valid t
_____________________________________(1/1)
exists t :  ftree, ftree_formula t = lor C (lor A B) ∧ valid t
```

Where the bar separates our hypothesis `H` from our goal on the bottom (the `(1/1)` indicates that we are on case 1 out of 1). Since Coq is constructive, from the existential statement we have we can extract a specific witness, and so calling `destruct H as [t H]` gives us an `ftree` called `t`, as well as the hypothesis `H` that it satisfies the conjunction. On the flipside, to prove our existential statement we need to produce a specific `ftree` witness, and `eapply ex_intro` sets us up to do that by removing the existential quantifier in place of a variable name we must instantiate. We can now use the `instantiate` command to construct this object (the `1:=` simply indicates we're instantiating the first (and in our case only) variable): start with `t`, the tree we already have by assumption, then apply `exchange_abd` to it, apply `exchange_cab` to the resulting tree, and in turn apply `exchange_ab` to this result, and we now an `ftree` which Coq recognizes (after a `simpl` and `auto`) is identical to the one we wanted.

Not including the existential management, this proof is structurally identical with our proof of `associativity1`. However, while there we could start with our proof and apply the 3 exchange rules in sequence step-by-step, here we have to apply them all at once, and backwards, so it is harder to construct longer derivations this way. We can see our intermediate steps if we make them as `assert` statements, and in fact this was how we originally constructed the above proof, but that takes somewhat longer than being able to apply our inference rules to our hypothesis directly. We cannot do this here, because once we discharge the existential quantifier with `destruct`, we now have a `t` and our hypothesis about it. We can do something like:

```
pose (exchange_abd C A B t) as t'.
```

Which would create the following in our proof window:

```
t' := exchange_abd C A B t :  ftree
```

But this does not really get us anywhere and this could be handled much better with an `assert`. Hence, for longer step-by-step derivations, which are the mainstay of the $PA \subseteq PA_\omega$ proof, our `PA_omega_theorem` is strongly preferred, and so we intend to use that and port its proofs over with `provable_theorem` above. Luckily, the rest of the consistency proof will not involve these, and we will mostly come down to making delicate proof substitutions, for which (as we will see in the next section) proof trees are well-suited for.

In preparation for this, the rest of the section proves some prerequisite lemmas. Namely, that all theorems are closed formulas, that syntactic equality of formulas is decidable, and that if we have a proof tree:

$$\frac{\begin{array}{cccc}\vdots & \vdots & \vdots & \\ A(0) & A(1) & A(2) & \dots\end{array}}{\forall m A(m)}\ \omega\text{-Rule}$$

Then for every m, the proof tree $g(m)$ is `valid` and has `ftree_formula` $A(m)$.

3.9 Invertibility Lemmas

The final 1000 lines of our code deals with Lemma 6 of chapter 2, and most particularly the substitution indicators we mentioned there. We do not know of any source that discusses the details of proof substitution even to the depth we did

in chapter 2. In fact, on first reading of Mendelson's [20] proof, we were scarcely conscious of the inescapable need to make non-local substitutions, and we even gave a full lecture in the Boise State Set Theory seminar on this proof where this gross omission was not even noticed. It was only in the process of writing chapter 2 that we came to see that understanding these global formula substitutions is probably the most important part of understanding this second half of the proof deeply, and so we studied Mendelson's proof with more care.

Even so, it was only after actually trying to implement this in Coq that we realized that we didn't fully understand how these proof substitution procedures work. Our discussion of all this in chapter 2 was only possible after we had actually struggled with this implementation for a few weeks. In particular, devising the data structure that eventually became substitution indicators was among the most difficult puzzles of this implementation, at least on par with the ordinal definitions and proof trees, if not greater.

From our discussion of Lemma 6 in chapter 2, it was relatively straightforward to implement the `ProofSub` procedure there, but only after implementing proof trees.[10] Handling our obstacle (b) discussed on page 64 also posed little obstacle with the machinery presented up to now, since our proof tree transformation procedure already had to be formally defined across all 19 cases anyways, and so making an exception for (e.g.) the Negation rule was easy to accommodate.

The heart of the matter was obstacle (a), mentioned on page 64, where we consider what happens if we have a proof:

$$\frac{\begin{array}{c}\vdots\\ \neg\neg B \vee \neg\neg B\end{array}}{}$$

[10]This was the part of the proof where we were forced into doing, and only after about a week of trying to make it work without changing the basic structure of `PA_omega_theorem`.

Then we want to transform this into a proof:

$$\dfrac{\vdots}{B \vee \neg\neg B}$$

But if we naïvely substitute B in place of $\neg\neg B$, we instead get a proof:

$$\dfrac{\vdots}{B \vee B}$$

Hence, we only want to make substitutions at certain places in our final formula. But this means we only want to make substitutions at certain places in the formulas above it. For instance, if our original proof tree looked like:

$$\dfrac{\dfrac{\vdots}{\neg\neg B \vee \neg\neg B}}{\neg\neg B \vee \neg\neg B}\ \text{Exchange}$$

Then we want to change it to:

$$\dfrac{\dfrac{\vdots}{\neg\neg B \vee B}}{B \vee \neg\neg B}\ \text{Exchange}$$

Because if we look at the first $\neg\neg B$ in the conclusion of the original proof, and "trace its path" up the proof tree, we see that this path goes through the *second* $\neg\neg B$ in the penultimate formula of the original proof. Mendelson did allude to this issue, and called this the *history* of (the first) $\neg\neg B$, although this was not rigorously defined. It is understandable why, since this *history* notion would have to be defined on a case-by-case basis with all the inference rules. For instance, we could also have:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\vdots}{(\neg\neg B \vee \neg\neg B) \vee (\neg\neg B \vee \neg\neg B) \vee \neg\neg B}}{(\neg\neg B \vee \neg\neg B) \vee \neg\neg B}\ \text{Contraction} \quad \dfrac{\vdots}{\neg\neg\neg B \vee \neg\neg B}}{(\neg\neg B \vee \neg\neg B) \vee \neg\neg B}\ \text{Cut}}{\neg\neg B \vee (\neg\neg B \vee \neg\neg B)}\ \text{Exchange}}{(\neg\neg B \vee \neg\neg B) \vee \neg\neg B}\ \text{Associativity}}{\neg\neg B \vee \neg\neg B}\ \text{Contraction}$$

In which case its easy to lose track of "where the first $\neg\neg B$ went" unless one has a very specific understanding of how to trace its path on a rule-by-rule basis. But even prior to this, suppose we could trace its history. That is, in the formula:

$$(\neg\neg B \vee \neg\neg B) \vee (\neg\neg B \vee \neg\neg B) \vee \neg\neg B$$

that shows up in the above proof tree, suppose we did know which $\neg\neg B$'s are in the history. How exactly do we implement the appropriate substitution?

Our ultimate solution was to define a new data structure called a *substitution indicator*, which consists of 0's and 1's in nested parentheses. These parentheses are meant to match up with the parentheses of a given formula, so that each 0 or 1 will match up with a subformula that is either atomic formula, negated, or quantified, but never a disjunction of formulas.[11] A "1" indicates that its corresponding subformula is a substitution target, while a "0" indicates it should be left alone. For instance, in the formula above, its substitution indicator should be of the form:

$$(b_1\ b_2)\ (b_3\ b_4)\ b_5$$

where every $b_i \in \{0, 1\}$. In particular (and as the reader can verify), in the above proof, this formula's substitution indicator will come out to:

$$(1\ 0)\ (1\ 0)\ 0$$

Then the resulting substitution would target the 1st and 3rd $\neg\neg B$, but not the

[11] For formulas like $\neg(C \vee D)$ or $\forall x(C \vee D)$, we thankfully never have to "reach inside" the negations or quantifiers in the substitutions we make, so these are always matched with a simple 0 or 1.

2nd, 4th, or 5th. Our implementation of this new structure is quite simple:

```
Inductive subst_ind :  Type :=
| ind_0 :  subst_ind
| ind_1 :  subst_ind
| lor_ind :  subst_ind -> subst_ind -> subst_ind.
Notation "(0)" := ind_0.
Notation "(1)" := ind_1.
Notation "( x y )" := (lor_ind x y).
```

Once we have this defined, we have to make sure that in substitutions over entire proof trees, we can track how the structure of a formula changes on a rule-by-rule basis, and change our `subst_ind` accordingly. In particular, in the proof tree example above, we want to have:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\vdots}{(1\ 0)\ (1\ 0)\ 0}}{(1\ 0)\ 0}\,\text{Contraction} \qquad \dfrac{\vdots}{0\ 1}}{(1\ 0)\ 1}\,\text{Cut}}{1\ (1\ 0)}\,\text{Exchange}}{(1\ 1)\ 0}\,\text{Associativity}}{1\ 0}\,\text{Contraction}$$

But we need more machinery to make this work. For one thing, even in just plain formula substitution, we need the substitution indicator to actually match the structure of our formula. So we define a boolean test for this:

```
Fixpoint subst_ind_fit (A : formula) (S : subst_ind) :  bool :=
match (A, S) with
| (lor B C, lor_ind S_B S_C) =>
    subst_ind_fit B S_B && subst_ind_fit C S_C
| (_, lor_ind _ _) => false
| (lor _ _, _) => false
| (_, _) => true
end.
```

Then our formula substitution function `formula_sub_ind` will first call this, doing nothing if there's a mismatch,[12] otherwise calling the function `formula_sub_ind_fit` which actually performs the substitution.

In the case of substitution over entire proofs, an immediate complication that arises is when new formulas get introduced as we move up the proof tree. For instance, in a case of Cut:

$$\dfrac{\dfrac{\vdots}{\neg\neg B \vee A} \quad \dfrac{\vdots}{\neg A \vee \neg\neg B}}{\neg\neg B \vee \neg\neg B}\ \text{Cut}$$

A might itself be a disjunction; indeed it may be of any structure possible. But either way, we do not want to substitute for *anything* in A no matter what it is. This motivates us to define:

[12] This will not actually happen in anything we are doing, but due to low-level technical details it has to be handled.

```
Fixpoint non_target (A : formula) :  subst_ind :=
match A with
| lor B C => lor_ind (non_target B) (non_target C)
| _ => (0)
end.
```

Which takes a formula A, and returns the `subst_ind` with the matching parenthetical structure whose binary values consist of all 0's. That way, we can track the substitution indicators in the previous Cut as follows:

$$\dfrac{\dfrac{\vdots}{1\ \texttt{(non\_target A)}} \quad \dfrac{\vdots}{0\ 0}}{1\ 0}\ \text{Cut}$$

The next 100 lines are simply proving some basic lemmas about this machinery, mainly the conditions when a `sub_ind` fits a given formula. After that, the remainder of this section is devoted to proving part (1) or Lemma 6, namely that if $PA_\omega \vdash \neg\neg B \vee D$, then $PA_\omega \vdash B \vee D$.

First, our task is to define our double negation substitution operation. Over formulas, this is just:

```
Definition dub_neg_sub_formula (A E : formula) (S : subst_ind) :  formula :=
    formula_sub_ind A (neg (neg E)) E S.
```

Over proof trees, the operation will be called `dub_neg_sub_ftree`, and this will first check to make sure the given `sub_ind` matches the `ftree_formula`, and if so,

it will call `dub_neg_sub_ftree_fit`. This takes in a proof tree $\mathscr{P}$, formula E, and substitution indicator S, and replaces $\neg\neg E$ with E as appropriate in the entire proof tree. At 118 lines, it is by far the longest definition in the file, and begins:

```
Fixpoint dub_neg_sub_ftree_fit
    (P : ftree) (E : formula) (S : subst_ind) :  ftree :=
match P, S with
| node A, _ => P
| exchange_ab A B P', lor_ind S_B S_A =>
    exchange_ab
        (dub_neg_sub_formula A E S_A)
        (dub_neg_sub_formula B E S_B)
        (dub_neg_sub_ftree_fit P' E (lor_ind S_A S_B))
```

In the `node` case, A is an axiom and so is not going to be $\neg\neg E$ anyways. For `exchange_ab`, our `ftree_formula` is $A \vee B$, so our `sub_ind` will be of the form `S_B S_A`, so we simply switch these and pass them along up the proof tree. Our case of Cut follows our discussion above:

```
| cut_cad C A D Q1 Q2, lor_ind S_C S_D =>
    cut_cad
        (dub_neg_sub_formula C E S_C)
        A
        (dub_neg_sub_formula D E S_D)
        (dub_neg_sub_ftree_fit Q1 E (lor_ind S_C (non_target A)))
        (dub_neg_sub_ftree_fit Q2 E (lor_ind (0) S_D))
```

Some of the rules are actually simple, since their conclusion cannot have $\neg\neg E$ anyways. For instance, the ω-Rule without a side formula is just:

```
| w_rule_a A n g, _ => P
```

On the other hand, with D it becomes rather interesting. The situation is:

$$\frac{\begin{array}{cccc}\vdots & \vdots & \vdots & \\ A(0)\vee D & A(1)\vee D & A(2)\vee D & \dots\end{array}}{(\forall x A(x))\vee D}\ \omega\text{-Rule}$$

Where D can be anything, and we might want to perform substitutions in D and its history. This means that the function `g` we have, which originally returned proofs of $A(n)\vee D$, needs to be modified so it returns proofs of this formula except with `dub_neg_sub_formula` applied to D. This results in:

```
| w_rule_ad A D n g, lor_ind S_A S_D =>
    w_rule_ad
        A
        (dub_neg_sub_formula D E S_D)
        n
        (fun (n :  nat) =>
            dub_neg_sub_ftree_fit (g n) E (lor_ind (non_target A) S_D))
```

The `fun` keyword is a classic functional programming construct which allows us to define anonymous functions. In this case, our expression on the last two lines defines the function which inputs n, and outputs the `ftree` that results from applying our proof tree substitution to $g(n)$ (and carrying around the substitution indicators properly).

Lastly, the Negation rule is the important special case we discussed in chapter 2. But here, deleting that node in the proof tree is straightforward:

```
| negation_a A P', _ =>
    (match eq_f A E, S with
    | true, (1) => P'
    | _, _ => P
    end)
```

Finally, after some more small lemmas, we prove the following:

```
Lemma dub_neg_ftree_formula' :  forall (P : ftree) (E : formula),
    valid P ->
    forall (S : subst_ind),
        subst_ind_fit (ftree_formula P) S = true ->
        ftree_formula (dub_neg_sub_ftree P E S) =
        dub_neg_sub_formula (ftree_formula P) E S.
```

Which took 142 lines, our longest proof so far, due to the 19 cases we had to cover when we inducted over `ftree`. Since the result holds trivially if the `subst_ind_fit` assumption is false, we easily obtain from this:

```
Lemma dub_neg_ftree_formula :  forall (P : ftree) (E : formula),
    valid P ->
    forall (S : subst_ind),
        ftree_formula (dub_neg_sub_ftree P E S) =
        dub_neg_sub_formula (ftree_formula P) E S.
```

Which we repeatedly use in the proof of the following:

```
Lemma dub_neg_valid :  forall (P : ftree) (E : formula),
    closed E = true -> valid P ->
    forall (S : subst_ind),
        subst_ind_fit (ftree_formula P) S = true ->
    valid (dub_neg_sub_ftree P E S).
```

which goes on for 377 lines. This also inducts over 19 cases, but also more subcases and subsubcases within those than the previous lemma. None of these repeated themselves enough to be amenable to the basic automation tactics this author is currently aware of, although there were certain patterns common enough that we can imagine the existence of such tactics. But with that in place, we relatively easily prove the final two theorems in the file:

```
Lemma double_negation_invertible_a :  forall (A : formula),
    provable (neg (neg A)) -> provable A.
Lemma double_negation_invertible_ad :  forall (A D : formula),
    provable (lor (neg (neg A)) D) -> provable (lor A D).
```

Remaining from section 4 of chapter 2, of course, are parts (2-3) of Lemma 6, which involve making the formula substitutions $[\neg B/\neg(B \vee C)]$ and $[B(n)/\forall x B(x)]$ instead of $[B/\neg\neg B]$. But as our proofs there indicate, there is nothing else different about these procedures besides the inference rule where we have to delete a node in the proof tree, and that was a trivial part of this implementation.

There is also Lemma 7 (Erasure Lemma), where we are actually erasing formulas

from a proof tree, and thus changing the structure of formulas as we go up. For this we will have to change a few things in our proof template, particularly how the substitution indicators will change on a rule-by-rule basis with these deletions, but we would be surprised if writing these definitions took more than a few minutes thought.

Lastly, Lemmas 6 and 7 also make claims about ordinals and degrees, which we do not have yet. However, there is no reason to believe our results in this section will not go through when we attach these, and we expect this to be a tedious but routine matter.

3.10 Cut-Elimination

There is no code here presently, but this is where we will do the actual Cut-elimination argument from section 5 of chapter 2. At this point, we will have both the invertibility lemmas from the previous section and the ordinal arithmetic machinery from much earlier. Here, we will actually have to do nontrivial operations on the ordinals attached to proof trees. While we don't expect this to be difficult, we cannot be completely certain on this point until we've actually decorated our proof trees and seen how the ordinals interact.

Otherwise, the other arguments here involve nothing we have not already done. Case (4) is the most involved, and will involve defining some new substitution procedures over proof trees, but nothing any harder than the previous section, and we expect to be able to use that template without major roadblocks.

3.11 Dangerous Disjuncts

Here is where we plan on making the high-level observations about the consistency of PA. From chapter 2, this is section 2 at the beginning of the proof, as well as Corollaries 3 and 4 at the very end. Here, we will have to define `consistent`, `dangerous_disjunction`, and `dangerous_rule`. This will take a modest amount of machinery we do not currently have, but we would claim that there is very little room that part of our discussion for any particularly difficult proofs to hide.

CHAPTER 4

CONCLUSION

Given our observations in Chapter 3, we estimate it will take about 2-3 more months to finish implementing our Chapter 2 proof in Coq. This will end up being substantially more than our current 5000 lines, mainly because there are many proofs we will have to do that are similar to the proof tree transformations described in section 3.9. These will all mostly follow the template of `dub_neg_valid`, but unfortunately each of these will still require distinct (and tedious) proofs.

Besides this, the main task is to attach degrees and ordinals to our proof trees. As we emphasized in chapter 3, we do not expect this to pose many significant challenges, but this probably will result in our proof trees being more cumbersome when we have to carry these decorations around in every computation. We are not sure exactly *how* much extra tedium this will introduce, and this is our main uncertainty in the above estimate.

However, there is a deeper problem with our implementation that the above does not include: the proof-theoretic strength of the Calculus of Constructions is much greater than $PRA + \epsilon_0$. Hence, once this implementation as described is finished, we cannot really claim to have verified:

$$PRA + \epsilon_0 \vdash \mathrm{Con}_{PA}$$

But instead we will have merely verified the (trivial and unsurprising) statement:

$$CoC \vdash \mathrm{Con}_{PA}$$

Of course, it is our claim that, even though we have had access to the powerful machinery of CoC, we have not actually *used it*, and everything we have done can be formalized in $PRA + \epsilon_0$. And we believe that any diligent reader who scrolls through our current 5000+ lines together with our 50+ page explanation of it in chapter 3 will agree with us. Nevertheless, this is still a *claim*, a claim that we have not verified.

We have not thought deeply about how to address this. We do have tentative ideas, which but these remarks should be taken to be preliminary and not well-researched:

1. In the worst-case scenario, we can use a different theorem prover besides Coq. In particular, we would like something in the "logical framework"[23] style (such as Isabelle) where the underlying theory is malleable, and consequently its proof-theoretic strength flexible. We have not used any theorem prover besides Coq, so we know little about how this works, but certainly we expect to be able to figure this out in relatively short order. At that point, it is unclear *how* easily we can transfer our code into this new prover, but certainly the concepts need not be reinvented, and so this should take much less time than it took to get to this point. We are also not certain if there are theorem provers in this style that can support theories as weak as what we want; PRA is *quite* weak in the context of mainstream mathematics outside of proof theory, and we know of very little work that involved computer-verifying anything of proof theory.

2. We can build the system $PRA + \epsilon_0$ within Coq, much as we built PA_ω (and expect to easily build PA), and then prove something akin to:

```
Theorem main_result :  PRA_e0_theorem Con_PA.
Proof.
pose proof PRA_e0_theorem Con_PA_omega.
    ⋮
```

The main drawback to this is, we would always be trying to "reason one level down", the difference in every theorem being:

```
Theorem theorem_name :  theorem_statement.
Theorem theorem_name :  PRA_e0_theorem theorem_statement.
```

And our experience with PA_ω suggests that it is usually much more awkward to prove `PA_omega_theorem A` than to prove `A` directly in Coq.

There is also a technical obstacle here in how to get PRA to represent proof trees in PA_ω. In our discussion of the ω-Rule that began on page 100, we pointed out that we can recast the rule to require a primitive recursive function g such that:

$$PRA \vdash \forall n \text{“}g(n) \text{ is a } PA_\omega \text{ proof of } A(n)\text{”}$$

The problem is, since PA_ω proofs are infinite, PRA cannot in general formulate the statement in quotes. We thank Jeremy Avigad for pointing this out to us, and suggesting continuous cut elimination as a potential workaround, but we have not yet had the time to pursue this matter further.

3. We noted above that, in the best-case scenario, we will simply *verify* that we have not done anything beyond $PRA + \epsilon_0$ in all 5000+ lines. Suppose there were a way to do this somehow. How would this work?

   There would have to be some Coq function, call it `RM`, that automatically does reverse mathematics: it can look at another Coq proof, tactic-by-tactic, and scrutinize what axioms it is using. Of course, actual mathematicians in reverse mathematics scrutinize *theorems* rather than specific *proofs*. That is to say, if a theorem was proved with strong axioms, they will try to come up with a new proof that uses weaker axioms, but proof discovery is a much harder problem than what we wish to tackle with this hypothetical tool. But reverse mathematicians also often spend time tediously scrutinizing existing proofs for which axioms are used, and in our view this process can plausibly be automated to a large degree.

   And ultimately, the extent of this matters, and `RM` could only (feasibly) give a relatively coarse upper bound on the axiomatic strength of a given proof. For instance, if the proof uses `induction` on the natural numbers, this is easy to detect, and it may well say that it is using the strength of PA with its induction schema. But then, it can also determine the number of alternating quantifiers in the formula inducted on, e.g. if its a Σ_1 formula, then we can say we are only using the power of $I\Sigma_1$. But there is also an infinite hierarchy of theories between Q and $I\Sigma_1$, starting with $I\Sigma_0$, and there is even an infinite hierarchy of theories even between Q and $I\Delta_0$ [42].

   The fact is, it would not be infeasible to do this in a way that will satisfy every proof theorist. However, it may be realistic to hard-code enough rules to roughly

assess proof-theoretic strength in most proofs, and furthermore, to locate tactics that apparently carry axiomatic heft. The example above illustrates how `RM` might do this in many cases. Other "black-box" tactics like `auto` may have to be actively dissected, but our impression is that these tend to do relatively simple things that can be formalized in nearly trivial systems.

Such a tool may not exist currently for us to use, but the above remarks suggest that it may be feasible to design one that can upper-bound the proof-theoretic strength of most of our code in one fell swoop, leaving scattered other parts which can be verified manually. In our estimation, this is very unlikely to be worth making for this specific case, but it may be more broadly applicable, since it could be applied to all existing Coq proofs, and in so doing perhaps yield a wealth of (very coarse-grained) reverse math facts for free.

With that all being said, we think it is not premature to reflect on what this implementation would mean once it is finished.

In developing his consistency proof, Gentzen heralded the beginning of ordinal analysis, introduced the now ubiquitous proof-theoretic technique of cut-elimination, and even reshaped our thinking of proof calculi in general, by developing the systems of natural deduction and sequent calculus in the process. For these reasons, we believe it is fair to consider his result the most significant in all proof theory, after Gödel's 2nd incompleteness theorem.

The latter had not been mechanized in a theorem prover until it was done in 2013 in Isabelle, as described in [21, 22], and to our knowledge this is the most comparable research project to ours. This was done by Larry Paulson, Isabelle's original developer, using the theory of hereditarily finite sets (HF) to make the Gödel

encoding more natural, and thereby proving both incompleteness theorems (the 1st had been done before in multiple provers). In future work we hope to study this–to the extent our knowledge of Isabelle syntax allows–to get a better understanding of how to arithmetize proof theory and represent proofs within proofs. According to slides from one of Paulson's presentations, the project took, "<9 months for the first theorem, a further 4 for the second",[1] and according to one of the resulting papers, "the machine proofs are fairly concise at under 12,400 lines for both theorems."

Another related project, also in Isabelle, is a 2004 mechanized proof of Gödel's completeness theorem [19]. This is a simpler theorem, with the code being just over 2000 lines, but has cut-elimination of pure FOL as a corollary. Hence, our project will technically not be the first mechanization of cut-elimination, but to our knowledge, it *is* the first mechanization of the class of techniques that proof theorists associate with the term "cut-elimination".

Moreover, we believe that this is important, given the detail-oriented nature of cut-elimination arguments. We have already noted how we failed to understand the subtleties of the cut-elimination step in chapter 2, even after we lectured about it to a seminar of logicians. But anecdotes aside, even many published cut-elimination arguments have later been shown incorrect [18, 12, 6]. Given the nature of cut-elimination, this is not surprising: to our understanding, there is no simple explanation "why it works"; rather it works because that is how a large mess of details happen to work the way they do (and this varies from system to sytem). The 4-color theorem, for instance, seems to have a *proof* but no *explanation*, where by the latter we mean (roughly) a small set of reasons that can help a human understand *why* a a theorem is true. In this regard, the 4-color theorem has no rival, but we feel that cut-elimination has a

[1] `https://www.cl.cam.ac.uk/~lp15/papers/Formath/Goedel-slides.pdf`

similar quality, and hence we cannot easily gain confidence in (particular uses of) it without some tool that excels at checking/verifying details.

Gentzen's result itself, of course, stood quite firmly on its own before our verification efforts, having been independently pored over in detail by 3 generations of proof theorists. Nevertheless, as we noted in section 1.11, the early consistency proofs, however interesting, suffered from counterexamples. So there is something to be said, however symbolically, for getting things right *this* time around.

REFERENCES

[1] John L. Bell. The axiom of choice. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, spring 2019 edition, 2019.

[2] Yves Bertot and Pierre Castéran. *Interactive theorem proving and program development*. Texts in Theoretical Computer Science. An EATCS Series. Springer-Verlag, Berlin, 2004. Coq'Art: the calculus of inductive constructions, With a foreword by Gérard Huet and Christine Paulin-Mohring.

[3] George Boole. *An investigation of the laws of thought: on which are founded the mathematical theories of logic and probabilities*. Dover Publications, 1854.

[4] Samuel R. Buss. First-order proof theory of arithmetic. In *Handbook of proof theory*, volume 137 of *Stud. Logic Found. Math.*, pages 79–147. North-Holland, Amsterdam, 1998.

[5] Timothy Y. Chow. The Consistency of Arithmetic. *Math. Intelligencer*, 41(1):22–30, 2019.

[6] Agata Ciabattoni and Kazushige Terui. Modular cut-elimination: finding proofs or counterexamples. In *Logic for programming, artificial intelligence, and reasoning*, volume 4246 of *Lecture Notes in Comput. Sci.*, pages 135–149. Springer, Berlin, 2006.

[7] William Ewald. The emergence of first-order logic. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, spring 2019 edition, 2019.

[8] José Ferreirós. The early development of set theory. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, fall 2016 edition, 2016.

[9] Gottlob Frege. *Die Grundlagen der Arithmetik: eine logisch mathematische Untersuchung über den Begriff der Zahl*. w. Koebner, 1884.

[10] Gottlob Frege. *Grundgesetze der arithmetik*, volume 1. H. Pohle, 1893.

[11] Georges Gonthier. A computer-checked proof of the four colour theorem.

[12] Rajeev Goré, Linda Postniece, and Alwen Tiu. Cut-elimination and proof-search for bi-intuitionistic logic using nested sequents. In *Advances in modal logic. Vol. 7*, pages 43–66. Coll. Publ., London, 2008.

[13] I. Grattan-Guinness. *The search for mathematical roots, 1870–1940*. Princeton Paperbacks. Princeton University Press, Princeton, NJ, 2000. Logics, set theories and the foundations of mathematics from Cantor through Russell to Gödel.

[14] David Hilbert. Mathematical problems. *Bull. Amer. Math. Soc.*, 8(10):437–479, 1902.

[15] David Hilbert. Über das Unendliche. *Math. Ann.*, 95(1):161–190, 1926.

[16] IamMeeoh. Understanding the countable ordinals up to ϵ_0. MathOverflow. URL: `https://mathoverflow.net/q/56062` (version: 2017-10-31).

[17] Thomas Jech. *Set theory*. Springer Monographs in Mathematics. Springer-Verlag, Berlin, 2003. The third millennium edition, revised and expanded.

[18] Ori Lahav and Arnon Avron. A semantic proof of strong cut-admissibility for first-order Gödel logic. *J. Logic Comput.*, 23(1):59–86, 2013.

[19] James Margetson and Tom Ridge. Completeness theorem. *Archive of Formal Proofs*, September 2004. `http://isa-afp.org/entries/Completeness.html`, Formal proof development.

[20] Elliott Mendelson. *Introduction to mathematical logic*. D. Van Nostrand Co., Inc., Princeton, N.J., 1964.

[21] Lawrence C. Paulson. A machine-assisted proof of Gödel's incompleteness theorems for the theory of hereditarily finite sets. *Rev. Symb. Log.*, 7(3):484–498, 2014.

[22] Lawrence C. Paulson. A mechanised proof of Gödel's incompleteness theorems using nominal Isabelle. *J. Automat. Reason.*, 55(1):1–37, 2015. `https://www.cl.cam.ac.uk/~lp15/papers/Formath/Goedel-ar.pdf`.

[23] Frank Pfenning. Logical frameworksa brief introduction. In *Proof and system-reliability*, pages 137–166. Springer, 2002.

[24] Benjamin C Pierce, Chris Casinghino, Marco Gaboardi, Michael Greenberg, Cătălin Hriţcu, Vilhelm Sjöberg, and Brent Yorgey. Software foundations, volume 1: Logical foundations. 2010. `http://www.cis.upenn.edu/bcpierce/sf/current/index.html`.

[25] Henri Poincaré and Francis Maitland. *Science and method*. Courier Corporation, 2003.

[26] Michael Rathjen. The art of ordinal analysis. In *International Congress of Mathematicians. Vol. II*, pages 45–69. Eur. Math. Soc., Zürich, 2006.

[27] Constance Reid. *Hilbert*. With an appreciation of Hilbert's mathematical work by Hermann Weyl. Springer-Verlag, New York-Berlin, 1970.

[28] Kurt Schütte. Beweistheoretische Erfassung der unerdlichen Induktion in der Zahlentheorie. *Math. Ann.*, 122:369–389, 1951.

[29] Noah Schweber. How do we know pa is incomparable with pra + ϵ_0? Mathematics Stack Exchange. URL: `https://math.stackexchange.com/q/3130662` (version: 2019-02-28).

[30] Wilfried Sieg. *Hilbert's programs and beyond*. Oxford University Press, Oxford, 2013.

[31] Morgan Sinclaire. How to define this dependently-typed tree structure in coq? Stack Overflow. URL: `https://stackoverflow.com/questions/52659420/how-to-define-this-dependently-typed-tree-structure-in-coq` (version: 2018-10-05).

[32] Peter Smith. Mendelson's *Mathematical Logic* and the missing appendix on the consistency of pa. Mathematics Stack Exchange. URL: `https://math.stackexchange.com/q/288117` (version: 2013-02-12).

[33] C. Smoryński. *Self-reference and modal logic*. Universitext. Springer-Verlag, New York, 1985.

[34] Ernst Snapper. The three crises in mathematics: logicism, intuitionism and formalism. *Math. Mag.*, 52(4):207–216, 1979.

[35] David Speyer. The technical part of godels proof. Secret Blogging Seminar. URL: `https://sbseminar.wordpress.com/2009/12/07/the-technical-part-of-godels-proof/`.

[36] William W Tait. Finitism. *The Journal of Philosophy*, 78(9):524–546, 1981.

[37] Alfred Tarski. *Undecidable theories*. Studies in Logic and the Foundations of Mathematics. North-Holland Publishing Company, Amsterdam, 1953. In collaboration with Andrzej Mostowski and Raphael M. Robinson.

[38] user4894. What are the origins of galileo's paradox? History of Science and Mathematics Stack Exchange. `https://hsm.stackexchange.com/questions/5712/what-are-the-origins-of-galileos-paradox`.

[39] Jean Van Heijenoort. *From Frege to Gödel: a source book in mathematical logic, 1879-1931*, volume 9. Harvard University Press, 1967.

[40] Jan von Plato. *Saved from the cellar*. Sources and Studies in the History of Mathematics and Physical Sciences. Springer, Cham, 2017. Gerhard Gentzen's shorthand notes on logic and foundations of mathematics.

[41] Alfred North Whitehead and Bertrand Russell. *Principia mathematica*, volume 2. University Press, 1912.

[42] George Wilmers. Bounded existential induction. *J. Symbolic Logic*, 50(1):72–90, 1985.

[43] Richard Zach. The practice of finitism: epsilon calculus and consistency proofs in Hilbert's program. *Synthese*, 137(1-2):211–259, 2003. History of logic (Helsinki, 2000).

[44] Richard Zach. Hilberts program then and now. *Philosophy of Logic*, 5:411–447, 2006.

[45] Edward N. Zalta. Gottlob frege. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, summer 2018 edition, 2018.