1-1-2015

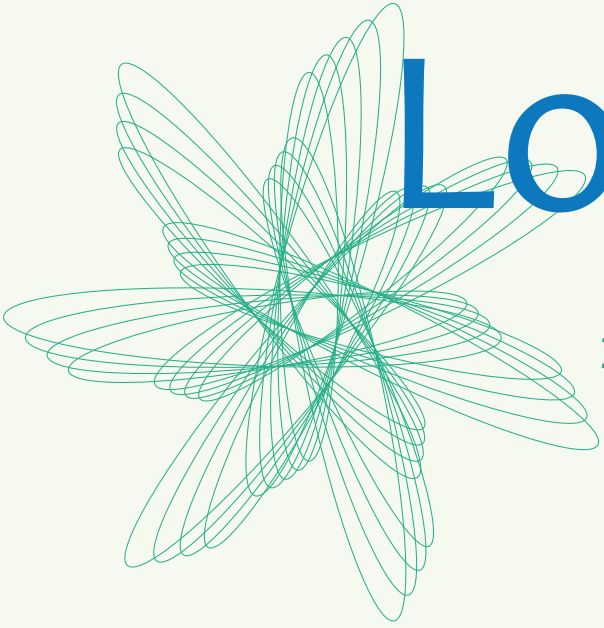
A Friendly Introduction to Mathematical Logic

Christopher C. Leary
SUNY Geneseo

Lars Kristiansen

A Friendly Introduction to

Mathematical Logic



2nd Edition

Christopher C. Leary
Lars Kristiansen

A Friendly Introduction to Mathematical Logic

A Friendly Introduction to Mathematical Logic

2nd Edition

Second printing
With corrections and some renumbered exercises

Christopher C. Leary

State University of New York
College at Geneseo

Lars Kristiansen

The University of Oslo

Milne Library, SUNY Geneseo, Geneseo, NY

ISBN: 978-1-942341-07-9

Milne Library
SUNY Geneseo
One College Circle

Geneseo, NY 14454

Lars Kristiansen has received financial support from the Norwegian Non-fiction Literature Fund

Contents

Preface

Preface to the First Edition

This book covers the central topics of first-order mathematical logic in a way that can reasonably be completed in a single semester. From the core ideas of languages, structures, and deductions we move on to prove the Soundness and Completeness Theorems, the Compactness Theorem, and Gödel's First and Second Incompleteness Theorems. There is an introduction to some topics in model theory along the way, but I have tried to keep the text tightly focused.

One choice that I have made in my presentation has been to start right in on the predicate logic, without discussing propositional logic first. I present the material in this way as I believe that it frees up time later in the course to be spent on more abstract and difficult topics. It has been my experience in teaching from preliminary versions of this book that students have responded well to this choice. Students have seen truth tables before, and what is lost in not seeing a discussion of the completeness of the propositional logic is more than compensated for in the extra time for Gödel's Theorem.

I believe that most of the topics I cover really deserve to be in a first course in mathematical logic. Some will question my inclusion of the Löwenheim–Skolem Theorems, and I freely admit that they are included mostly because I think they are so neat. If time presses you, that section might be omitted. You may also want to soft-pedal some of the more technical results in Chapter 5.

The list of topics that I have slighted or omitted from the book is depressingly large. I do not say enough about recursion theory or model theory. I say nothing about linear logic or modal logic or second-order logic. All of these topics are interesting and important, but I believe that they are best left to other courses. One semester is, I believe, enough time to cover the material outlined in this book relatively thoroughly and at a reasonable pace for the student.

Thanks for choosing my book. I would love to hear how it works for you.

To the Student

Welcome! I am really thrilled that you are interested in mathematical logic and that we will be looking at it together! I hope that my book will serve you well and will help to introduce you to an area of mathematics that I have found fascinating and rewarding.

Mathematical logic is absolutely central to mathematics, philosophy, and advanced computer science. The concepts that we discuss in this book—models and structures, completeness and incompleteness—are used by mathematicians in every branch of the subject. Furthermore, logic provides a link between mathematics and philosophy, and between mathematics and theoretical computer science. It is a subject with increasing applications and of great intrinsic interest.

One of the tasks that I set for myself as I wrote this book was to be mindful of the audience, so let me tell you the audience that I am trying to reach with this book: third- or fourth-year undergraduate students, most likely mathematics students. The student I have in mind may not have taken very many upper-division mathematics courses. He or she may have had a course in linear algebra, or perhaps a course in discrete mathematics. Neither of these courses is a prerequisite for understanding the material in this book, but some familiarity with proving things will be required.

In fact, you don't need to know very much mathematics at all to follow this text. So if you are a philosopher or a computer scientist, you should not find any of the core arguments beyond your grasp. You do, however, have to work abstractly on occasion. But that is hard for all of us. My suggestion is that when you are lost in a sea of abstraction, write down three examples and see if they can tell you what is going on.

At several points in the text there are asides that are indented and start with the word *Chaff*. I hope you will find these comments helpful. They are designed to restate difficult points or emphasize important things that may get lost along the way. Sometimes they are there just to break up the exposition. But these asides really are chaff, in the sense that if they were blown away in the wind, the mathematics that is left would be correct and secure. But do look at them—they are supposed to make your life easier.

Just like every other math text, there are exercises and problems for you to work out. Please try to at least think about the problems. Mathematics is a contact sport, and until you are writing things down and trying to use and apply the material you have been studying, you don't really know the subject. I have tried to include problems of different levels of difficulty, so some will be almost trivial and others will give you a chance to show off.

This is an elementary textbook, but elementary does not mean easy. It was not easy when we learned to add, or read, or write. You will find the going tough at times as we work our way through some very difficult and technical results. But the major theorems of the course—Gödel's Com-

pleteness Theorem, the incompleteness results of Gödel and Rosser, the Compactness Theorem, the Löwenheim–Skolem Theorem—provide wonderful insights into the nature of our subject. What makes the study of mathematical logic worthwhile is that it exposes the core of our field. We see the strength and power of mathematics, as well as its limitations. The struggle is well worth it. Enjoy the ride and see the sights.

Thanks

Writing a book like this is a daunting process, and this particular book would never have been produced without the help of many people. Among my many teachers and colleagues I would like to express my heartfelt thanks to Andreas Blass and Claude Laflamme for their careful readings of early versions of the book, for the many helpful suggestions they made, and for the many errors they caught.

I am also indebted to Paul Bankston of Marquette University, William G. Farris of the University of Arizona at Tucson, and Jiping Liu of the University of Lethbridge for their efforts in reviewing the text. Their thoughtful comments and suggestions have made me look smarter and made my book much better.

The Department of Mathematics at SUNY Geneseo has been very supportive of my efforts, and I would also like to thank the many students at Oberlin and at Geneseo who have listened to me lecture about logic, who have challenged me and rewarded me as I have tried to bring this field alive for them. The chance to work with undergraduates was what brought me into this field, and they have never (well, hardly ever) disappointed me.

Much of the writing of this book took place when I was on sabbatical during the fall semester of 1998. The Department of Mathematics and Statistics at the University of Calgary graciously hosted me during that time so I could concentrate on my writing.

I would also like to thank Michael and Jim Henle. On September 10, 1975, Michael told a story in Math 13 about a barber who shaves every man in his town that doesn't shave himself, and that story planted the seed of my interest in logic. Twenty-two years later, when I was speaking with Jim about my interest in possibly writing a textbook, he told me that he thought that I should approach my writing as a creative activity, and if the book was in me, it would come out well. His comment helped give me the confidence to dive into this project.

The typesetting of this book depended upon the existence of Leslie Lamport's LaTeX. I thank everyone who has worked on this typesetting system over the years, and I owe a special debt to David M. Jones for his Index package, and to Piet von Oostrum for Fancyheadings.

Many people at Prentice Hall have worked very hard to make this book a reality. In particular, George Lobell, Gale Epps, and Lynn Savino have

been very helpful and caring. You would not be holding this book without their efforts.

But most of all, I would like to thank my wife, Sharon, and my children, Heather and Eric. Writing this book has been like raising another child. But the real family and the real children mean so much more.

Preface to the Second Edition

From Chris:

I was very happy with the reaction to the first edition of **A Friendly Introduction**. I heard from many readers with comments, errors (both small and embarrassingly large), and requests for solutions to the exercises. The many kind words and thoughtful comments were and are much appreciated, and most, if not all, of your suggestions have been incorporated into the work you have before you. Thank you all!

As is often the case in publishing ventures, after a while the people at Prentice-Hall thought that the volume of sales of my book was not worth it to them, so they took the book out of print and returned the rights to me. I was very pleased when I received an email from Lars Kristiansen in September of 2012 suggesting that we work together on a second edition of the text and with the idea of including a section on computability theory as well as solutions to some of the exercises, solutions that he had already written up. This has allowed us to chart two paths to the incompleteness theorems, splitting after the material in Chapter 4. Readers of the first edition will find that the exposition in Chapters 5 and 6 follows a familiar route, although the material there has been pretty thoroughly reworked. It is also possible, if you choose, to move directly from Chapter 4 to Chapter 7 and see a development of computability theory that covers the Entscheidungsproblem, Hilbert's 10th Problem, and Gödel's First Incompleteness Theorem.

I am more than happy to have had the chance to work with Lars on this project for the last couple of years, and to have had his careful and creative collaboration. Lars has added a great deal to the work and has improved it in many ways. I am also in debt to the Department of Mathematics at the University of Oslo for hosting me in Norway during a visit in 2013 so that Lars and I could work on the revision face-to-face.

The staff at Milne Library of SUNY Geneseo have been most helpful and supportive as we have moved toward bringing this second edition to fruition. In particular, Cyril Oberlander, Katherine Pitcher, and Allison Brown have been encouraging and comforting as we have worked through the details of publication and production.

As in the first edition, I mostly have to thank my family. Eric and Heather, you were two and five when the first edition came out. I don't think either of you will read this book, even now, but I hope you know that

you are still my most important offspring. And Sharon, thanks to you for all of your support and love. Also thanks for taking one for the team and accompanying me to Oslo when I had to work with Lars. I know what a sacrifice that was.

This edition of the book is much longer than the original, and I am confident that it is a whole lot better. But the focus of the book has not changed: Lars and I believe that we have outlined an introduction to important areas of mathematical logic, culminating in the Incompleteness Theorems, that can reasonably be covered in a one-semester upper division undergraduate course. We hope that you agree!

From Lars:

First of all, I will say thank you to Chris for letting me in on this project. We have worked very well together and complemented each other in a number of respects.

I should also express my thanks to those who through the years have shaped my academic taste and pursuits. They have in some sense contributed to this book. Among them you find my teachers, colleagues and students at the University of Oslo. I cannot mention them all – I can probably not even remember them all – but a few names that immediately come to my mind are Stål Aanderaa, Herman Ruge Jervell (my PhD supervisor), Dag Normann, and Mathias Barra.

Finally, I will like to thank Dag Normann and Amir Ben-Amram for discussions and helpful comments on early versions of Chapter 7.

Our target group is undergraduate students that have reached a certain level of mathematical maturity but do not know much formal logic – maybe just some propositional logic – maybe nothing. It is the needs of the readers in this group that we want to meet, and we have made our efforts to do so: We have provided exercises of all degrees of difficulty, and we have provided detailed solutions to quite a few of them. We have provided discussions and explanations that might prevent unnecessary misunderstandings. We have stuck to topics that should be of interest to the majority of our target group. We have tried to motivate our definitions and theorems . . . and we have done a number of other things that hopefully will help an undergraduate student that wants to learn mathematical logic.

This book conveys some of the main insights from what we today call classic mathematical logic. We tend to associate the word "classic" with something old. But the theorems in this book are not old. Not if we think about the pyramids. Neither if we think about Pythagoras, Euclid, and Diophantus – or even Newton and Leibniz. All the theorems in this book were conceived after my grandparents were born, some of them even after I was born. They are insights won by the past few generations. Many things that seem very important to us today will be more or less forgotten in a hundred years or so. The essence of classic mathematical logic will be

passed on from generation to generation as long as the human civilization exists. So, in some sense, this is a book for the future.

I dedicate this book to the coming generations and, in particular, to my seven-year-old daughter Mille.

Chapter 1

Structures and Languages

Let us set the stage. In the middle of the nineteenth century, questions concerning the foundations of mathematics began to appear. Motivated by developments in geometry and in calculus, and pushed forward by results in set theory, mathematicians and logicians tried to create a system of axioms for mathematics, in particular, arithmetic. As systems were proposed, notably by the German mathematician Gottlob Frege, errors and paradoxes were discovered. So other systems were advanced.

At the International Congress of Mathematicians, a meeting held in Paris in 1900, David Hilbert proposed a list of 23 problems that the mathematical community should attempt to solve in the upcoming century. In stating the second of his problems, Hilbert said:

> *But 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 [of arithmetic]:* To prove that they are not contradictory, that is, that a finite number of logical steps based upon them can never lead to contradictory results. *(Quoted in [Feferman 98])*

In other words, Hilbert challenged mathematicians to come up with a set of axioms for arithmetic that were guaranteed to be consistent, guaranteed to be paradox-free.

In the first two decades of the twentieth century, three major schools of mathematical philosophy developed. The Platonists held that mathematical objects had an existence independent of human thought, and thus the job of mathematicians was to discover the truths about these mathematical objects. Intuitionists, led by the Dutch mathematician L. E. J. Brouwer,

held that mathematics should be restricted to concrete operations performed on finite structures. Since vast areas of modern mathematics depended on using infinitary methods, Brouwer's position implied that most of the mathematics of the previous 3000 years should be discarded until the results could be reproved using finitistic arguments. Hilbert was appalled at this suggestion and he became the leading exponent of the Formalist school, which held that mathematics was nothing more than the manipulation of meaningless symbols according to certain rules and that the consistency of such a system was nothing more than saying that the rules prohibited certain combinations of the symbols from occurring.

Hilbert developed a plan to refute the Intuitionist position that most of mathematics was suspect. He proposed to prove, using finite methods that the Intuitionists would accept, that all of classical mathematics was consistent. By using finite methods in his consistency proof, Hilbert was sure that his proof would be accepted by Brouwer and his followers, and then the mathematical community would be able to return to what Hilbert considered the more important work of advancing mathematical knowledge. In the 1920s many mathematicians became actively involved in Hilbert's project, and there were several partial results that seemed to indicate that Hilbert's plan could be accomplished. Then came the shock.

On Sunday, September 7, 1930, at the Conference on Epistemology of the Exact Sciences held in Königsberg, Germany, a 24-year-old Austrian mathematician named Kurt Gödel announced that he could show that there is a sentence such that the sentence is true but not provable in a formal system of classical mathematics. In 1931 Gödel published the proof of this claim along with the proof of his Second Incompleteness Theorem, which said that no consistent formal system of mathematics could prove its own consistency. Thus Hilbert's program was impossible, and there would be no finitistic proof that the axioms of arithmetic were consistent.

Mathematics, which had reigned for centuries as the embodiment of certainty, had lost that role. Thus we find ourselves in a situation where we cannot prove that mathematics is consistent. Although we believe in our hearts that mathematics is consistent, we know in our brains that we will not be able to prove that fact, unless we are wrong. For if we are wrong, mathematics is inconsistent. And (as we will see) if mathematics is inconsistent, then it can prove anything, including the statement which says that mathematics is consistent.

So do we throw our hands in the air and give up the study

of mathematics? Of course not! Mathematics is still useful, it is still beautiful, and it is still interesting. It is an intellectual challenge. It compels us to think about great ideas and difficult problems. It is a wonderful field of study, with rewards for us all. What we have learned from the developments of the nineteenth and twentieth centuries is that we must temper our hubris. Although we can still agree with Gauss, who said that, "Mathematics is the Queen of the Sciences..." she no longer can claim to be a product of an immaculate conception.

Our study of mathematical logic will take us to a point where we can understand the statement and the proof of Gödel's Incompleteness Theorems. On our way there, we will study formal languages, mathematical structures, and a certain deductive system. The type of thinking, the type of mathematics that we will do, may be unfamiliar to you, and it will probably be tough going at times. But the theorems that we will prove are among the most revolutionary mathematical results of the twentieth century. So your efforts will be well rewarded. Work hard. Have fun.

1.1 Naïvely

Let us begin by talking informally about mathematical structures and mathematical languages.

There is no doubt that you have worked with mathematical models in several previous mathematics courses, although in all likelihood it was not pointed out to you at the time. For example, if you have taken a course in linear algebra, you have some experience working with $\mathbb{R}^2$, $\mathbb{R}^3$, and $\mathbb{R}^n$ as examples of vector spaces. In high school geometry you learned that the plane is a "model" of Euclid's axioms for geometry. Perhaps you have taken a class in abstract algebra, where you saw several examples of groups: The integers under addition, permutation groups, and the group of invertible $n \times n$ matrices with the operation of matrix multiplication are all examples of groups—they are "models" of the group axioms. All of these are mathematical models, or structures. Different structures are used for different purposes.

Suppose we think about a particular mathematical structure, for example $\mathbb{R}^3$, the collection of ordered triples of real numbers. If we try to do plane Euclidean geometry in $\mathbb{R}^3$, we fail miserably, as (for example) the parallel postulate is false in this structure. On the other hand, if we want to do linear algebra in $\mathbb{R}^3$, all is well and good, as we can think of the points of $\mathbb{R}^3$ as vectors and let the scalars be real numbers. Then the axioms for a real vector space are all true when interpreted in $\mathbb{R}^3$. We will say that $\mathbb{R}^3$

is a model of the axioms for a vector space, whereas it is not a model for Euclid's axioms for geometry.

As you have no doubt noticed, our discussion has introduced two separate types of things to worry about. First, there are the mathematical models, which you can think of as the mathematical worlds, or constructs. Examples of these include $\mathbb{R}^3$, the collection of polynomials of degree 17, the set of 3×2 matrices, and the real line. We have also been talking about the axioms of geometry and vector spaces, and these are something different. Let us discuss those axioms for a moment.

Just for the purposes of illustration, let us look at some of the axioms which state that V is a real vector space. They are listed here both informally and in a more formal language:

Vector addition is commutative: $(\forall u \in V)(\forall v \in V)u + v = v + u$.

There is a zero vector: $(\exists 0 \in V)(\forall v \in V)v + 0 = v$.

One times anything is itself: $(\forall v \in V)1v = v$.

Don't worry if the formal language is not familiar to you at this point; it suffices to notice that there *is* a formal language. But do let us point out a few things that you probably accepted without question. The addition sign that is in the first two axioms is not the same plus sign that you were using when you learned to add in first grade. Or rather, it *is* the same sign, but you *interpret* that sign differently. If the vector space under consideration is $\mathbb{R}^3$, you know that as far as the first two axioms up there are concerned, addition is vector addition. Similarly, the 0 in the second axiom is not the real number 0; rather, it is the zero vector. Also, the multiplication in the third axiom that is indicated by the juxtaposition of the 1 and the v is the scalar multiplication of the vector space, not the multiplication of third grade.

So it seems that we have to be able to look at some symbols in a particular formal language and then take those symbols and relate them in some way to a mathematical structure. Different interpretations of the symbols will lead to different conclusions as regards the truth of the formal statement. For example, if we take the commutivity axiom above and work with the space V being $\mathbb{R}^3$ but interpret the sign $+$ as standing for cross product instead of vector addition, we see that the axiom is no longer true, as cross product is not commutative.

These, then, are our next objectives: to introduce formal languages, to give an official definition of a mathematical structure, and to discuss truth in those structures. Beauty will come later.

1.2 Languages

We will be constructing a very restricted formal language, and our goal in constructing that language will be to be able to form certain statements about certain kinds of mathematical structures. For our work, it will be necessary to be able to talk about constants, functions, and relations, and so we will need symbols to represent them.

> *Chaff:* Let us emphasize this once more. Right now we are discussing the *syntax* of our language, the marks on the paper. We are not going to worry about the semantics, or meaning, of those marks until later—at least not formally. But it is silly to pretend that the intended meanings do not drive our choice of symbols and the way in which we use them. If we want to discuss left-hemi-semi-demi-rings, our formal language should include the function and relation symbols that mathematicians in this lucrative and exciting field customarily use, not the symbols involved in chess, bridge, or right-hemi-semi-para-fields. It is not our goal to confuse anyone more than is necessary. So you should probably go through the exercise right now of taking a guess at a reasonable language to use if our intended field of discussion was, say, the theory of the natural numbers. See Exercise 1.

Definition 1.2.1. A **first-order language** $\mathcal{L}$ is an infinite collection of distinct symbols, no one of which is properly contained in another, separated into the following categories:

1. *Parentheses:* (,).
2. *Connectives:* $\vee$, $\neg$.
3. *Quantifier:* $\forall$.
4. *Variables, one for each positive integer n:* $v_1, v_2, \ldots, v_n, \ldots$. The set of variable symbols will be denoted *Vars*.
5. *Equality symbol:* $=$.
6. *Constant symbols:* Some set of zero or more symbols.
7. *Function symbols:* For each positive integer n, some set of zero or more n-ary function symbols.
8. *Relation symbols:* For each positive integer n, some set of zero or more n-ary relation symbols.

To say that a function symbol is n-ary (or has arity n) means that it is intended to represent a function of n variables. For example, $+$ has arity 2. Similarly, an n-ary relation symbol will be intended to represent a relation on n-tuples of objects. This will be made formal in Definition 1.6.1.

To specify a language, all we have to do is determine which, if any, constant, function, and relation symbols we wish to use. Many authors, by the way, let the equality symbol be optional, or treat the equality symbol as an ordinary binary (i.e., 2-ary) relation symbol. We will assume that each language has the equality symbol, unless specifically noted.

> *Chaff:* We ought to add a word about the phrase "no one of which is properly contained in another," which appears in this definition. We have been quite vague about the meaning of the word *symbol*, but you are supposed to be thinking about marks made on a piece of paper. We will be constructing sequences of symbols and trying to figure out what they mean in the next few pages, and by not letting one symbol be contained in another, we will find our job of interpreting sequences to be much easier.
>
> For example, suppose that our language contained both the constant symbol ♡ and the constant symbol ♡♡ (notice that the first symbol is properly contained in the second). If you were reading a sequence of symbols and ran across ♡♡, it would be impossible to decide if this was one symbol or a sequence of two symbols. By not allowing symbols to be contained in other symbols, this type of confusion is avoided, leaving the field open for other types of confusion to take its place.

Example 1.2.2. Suppose that we were taking an abstract algebra course and we wanted to specify the language of groups. A group consists of a set and a binary operation that has certain properties. Among those properties is the existence of an identity element for the operation. Thus, we could decide that our language will contain one constant symbol for the identity element, one binary operation symbol, and no relation symbols. We would get

$$\mathcal{L}_G \text{ is } \{0, +\},$$

where 0 is the constant symbol and $+$ is a binary function symbol. Or perhaps we would like to write our groups using the operation as multiplication. Then a reasonable choice could be

$$\mathcal{L}_G \text{ is } \{1, {}^{-1}, \cdot\},$$

which includes not only the constant symbol 1 and the binary function symbol $\cdot$, but also a unary (or 1-ary) function symbol $^{-1}$, which is designed to pick out the inverse of an element of the group. As you can see, there is a fair bit of choice involved in designing a language.

Example 1.2.3. The language of set theory is not very complicated at all. We will include one binary relation symbol, $\in$, and that is all:

$$\mathcal{L}_{ST} \text{ is } \{\in\}.$$

The idea is that this symbol will be used to represent the elementhood relation, so the interpretation of the string $x \in y$ will be that the set x is an element of the set y. You might be tempted to add other relation symbols, such as $\subset$, or constant symbols, such as $\emptyset$, but it will be easier to define such symbols in terms of more primitive symbols. Not easier in terms of readability, but easier in terms of proving things about the language.

In general, to specify a language we need to list the constant symbols, the function symbols, and the relation symbols. There can be infinitely many [in fact, uncountably many (cf. the Appendix)] of each. So, here is a specification of a language:

$$\mathcal{L} \text{ is } \{c_1, c_2, \ldots, f_1^{a(f_1)}, f_2^{a(f_2)}, \ldots, R_1^{a(R_1)}, R_2^{a(R_2)}, \ldots\}.$$

Here, the c_i's are the constant symbols, the $f_i^{a(f_i)}$'s are the function symbols, and the $R_i^{a(R_i)}$'s are the relation symbols. The superscripts on the function and relation symbols indicate the arity of the associated symbols, so a is a mapping that assigns a natural number to a string that begins with an f or an R, followed by a subscripted ordinal. Thus, an official function symbol might look like this:

$$f_{17}^{223},$$

which would say that the function that will be associated with the 17th function symbol is a function of 223 variables. Fortunately, such dreadful detail will rarely be needed. We will usually see only unary or binary function symbols and the arity of each symbol will be stated once. Then the authors will trust that the context will remind the patient reader of each symbol's arity.

1.2.1 Exercises

1. Carefully write out the symbols that you would want to have in a language $\mathcal{L}$ that you intend to use to write statements of elementary algebra. Indicate which of the symbols are constant symbols, and the arity of the function and relation symbols that you choose. Now write out another language, $\mathcal{M}$ (i.e., another list of symbols) with the same number of constant symbols, function symbols, and relation symbols that you would *not* want to use for elementary algebra. Think about the value of good notation.

2. What are good examples of unary (1-ary) functions? Binary functions? Can you find natural examples of relations with arity 1, 2, 3, and 4? As

you think about this problem, stay mindful of the difference between the function and the function symbol, between the relation and the relation symbol.

3. In the town of Sneezblatt there are three eating establishments: McBurgers, Chez Fancy, and Sven's Tandoori Palace. Think for a minute about statements that you might want to make about these restaurants, and then write out $\mathcal{L}$, the formal language for your theory of restaurants. Have fun with this, but try to include both function and relation symbols in $\mathcal{L}$. What interpretations are you planning for your symbols?

4. You have been put in charge of drawing up the schedule for a basketball league. This league involves eight teams, each of which must play each of the other seven teams exactly two times: once at home and once on the road. Think of a reasonable language for this situation. What constants would you need? Do you need any relation symbols? Function symbols? It would be nice if your finished schedule did not have any team playing two games on the same day. Can you think of a way to state this using the formal symbols that you have chosen? Can you express the sentence which states that each team plays every other team exactly two times?

5. Let's work out a language for elementary trigonometry. To get you started, let us suggest that you start off with *lots* of constant symbols—one for each real number. It is tempting to use the symbol 7 to stand for the number seven, but this runs into problems. (Do you see why this is illegal? 7, 77, 7/3, ...) Now, what functions would you like to discuss? Think of symbols for them. What are the arities of your function symbols? Do not forget that you need symbols for addition and multiplication! What relation symbols would you like to use?

6. A computer language is another example of a language. For example, the symbol := might be a binary function symbol, where the interpretation of the instruction

$$x := 7$$

would be to alter the internal state of the computer by placing the value 7 into the position in memory referenced by the variable x. Think about the function associated with the binary function symbol

$$\text{if } \_\_\_\_\_, \text{ then } \_\_\_\_\_.$$

What are the inputs into this function? What sort of thing does the function do? Look at the statement

$$\text{If } x + y > 3, \text{ then } z := 7.$$

Identify the function symbols, constant symbols, and relation symbols. What are the arities of each function and relation symbol?

7. What would be a good language for the theory of vector spaces? This problem is slightly more difficult, as there are two different varieties of objects, scalars and vectors, and you have to be able to tell them apart. Write out the axioms of vector spaces in your language. Or, better yet, use a language that includes a unary function symbol for each real number so that scalars don't exist as objects at all!

8. It is not actually necessary to include function symbols in the language, since a function is just a special kind of relation. Just to see an example, think about the function $f : \mathbb{N} \to \mathbb{N}$ defined by $f(x) = x^2$. Remembering that a relation on $\mathbb{N} \times \mathbb{N}$ is just a set of ordered pairs of natural numbers, find a relation R on $\mathbb{N} \times \mathbb{N}$ such that (x, y) is an element of R if and only if $y = f(x)$. Convince yourself that you could do the same for any function defined on any domain. What condition must be true if a relation R on $A \times B$ is to be a function mapping A to B?

1.3 Terms and Formulas

Suppose that $\mathcal{L}$ is the language $\{0, +, <\}$, and we are going to use $\mathcal{L}$ to discuss portions of arithmetic. If we were to write down the string of symbols from $\mathcal{L}$,

$$(v_1 + 0) < v_1,$$

and the string

$$v_{17})(\forall + +(((0,$$

you would probably agree that the first string conveyed some meaning, even if that meaning were incorrect, while the second string was meaningless. It is our goal in this section to carefully define which strings of symbols of $\mathcal{L}$ we will use. In other words, we will select the strings that will have meaning.

Now, the point of having a language is to be able to make statements about certain kinds of mathematical systems. Thus, we will want the statements in our language to have the ability to refer to objects in the mathematical structures under consideration. So we will need some of the strings in our language to refer to those objects. Those strings are called the terms of $\mathcal{L}$.

Definition 1.3.1. If $\mathcal{L}$ is a language, a **term of** $\mathcal{L}$ is a nonempty finite string t of symbols from $\mathcal{L}$ such that either:

1. t is a variable, or
2. t is a constant symbol, or
3. $t :\equiv ft_1t_2 \dots t_n$, where f is an n-ary function symbol of $\mathcal{L}$ and each of the t_i is a term of $\mathcal{L}$.

A couple of things about this definition need to be pointed out. First, there is the symbol $:\equiv$ in the third clause. The symbol $:\equiv$ is *not* a part of the language $\mathcal{L}$. Rather it is a meta-linguistic symbol that means that the strings of $\mathcal{L}$-symbols on each side of the $:\equiv$ are identical. Probably the best natural way to read clause 3. would be to say that "t is $ft_1t_2\dots t_n$."

The other thing to notice about Definition 1.3.1 is that this is a definition by recursion, since in the third clause of the definition, t is a term if it contains substrings that are terms. Since the substrings of t are shorter (contain fewer symbols) than t, and as none of the symbols of $\mathcal{L}$ are made up of other symbols of $\mathcal{L}$, this causes no problems.

Example 1.3.2. Let $\mathcal{L}$ be the language $\{\overline{0}, \overline{1}, \overline{2}, \dots, +, \cdot\}$, with one constant symbol for each natural number and two binary function symbols. Here are some of the terms of $\mathcal{L}$: $\overline{714}$, $+\overline{3}\,\overline{2}$, $\cdot + \overline{3}\,\overline{2}\,\overline{4}$. Notice that $\overline{1}\,\overline{2}\,\overline{3}$ is not a term of $\mathcal{L}$, but rather is a sequence of three terms in a row.

> *Chaff:* The term $+\overline{3}\,\overline{2}$ looks pretty annoying at this point, but we will use this sort of notation (called *Polish notation*) for functions rather than the infix notation $(\overline{3} + \overline{2})$ that you are used to. We are not really being that odd here: You have certainly seen some functions written in Polish notation: $\sin(x)$ and $f(x, y, z)$ come to mind. We are just being consistent in treating addition in the same way. What makes it difficult is that it is hard to remember that addition really is just another function of two variables. But we are sure that by the end of this book, you will be very comfortable with that idea and with the notation that we are using.

A couple of points are probably worth emphasizing, just this once. Notice that in the application of the function symbols, there are no parentheses and no commas. Also notice that all of our functions are written with the operator on the left. So instead of $\overline{3} + \overline{2}$, we write $+\overline{3}\,\overline{2}$. The reason for this is for consistency and to make sure that we can parse our expressions.

Let us give an example. Suppose that, in some language or other, we wrote down the string of symbols $\heartsuit ¥ \uparrow \diamondsuit \# \# \int$. Assume that two of our colleagues, Humphrey and Ingrid, were waiting in the hall while we wrote down the string. If Humphrey came into the room and announced that our string was a 3-ary function symbol followed by three terms, whereas Ingrid proclaimed that the string was really a 4-ary relation symbol followed by two terms, this would be rather confusing. It would be *really* confusing if they were both correct! So we need to make sure that the strings that we write down can be interpreted in only one way. This property, called *unique readability*, is addressed in Exercise 7 of Section 1.4.1.

> *Chaff:* Unique readability is one of those things that, in the opinion of the authors, is important to know, interesting to

prove, and boring to read. Thus the proof is placed in (we do not mean "relegated to") the exercises.

Suppose that we look more carefully at the term $\cdot + \overline{3}\,\overline{2}\,\overline{4}$. Assume for now that the symbols in this term are supposed to be interpreted in the usual way, so that $\cdot$ means multiply, $+$ means add, and $\overline{3}$ means three. Then if we add some parentheses to the term in order to clarify its meaning, we get

$$\cdot(+\overline{3}\,\overline{2})\,\overline{4},$$

which ought to have the same meaning as $\cdot\overline{5}\,\overline{4}$, which is $\overline{20}$, just as you suspected.

Rest assured that we will continue to use infix notation, commas, and parentheses as seem warranted to increase the readability (by humans) of this text. So $ft_1t_2\ldots t_n$ will be written $f(t_1, t_2, \ldots, t_n)$ and $+\overline{3}\,\overline{2}$ will be written $\overline{3} + \overline{2}$, with the understanding that this is shorthand and that our official version is the version given in Definition 1.3.1.

The terms of $\mathcal{L}$ play the role of the nouns of the language. To make meaningful mathematical statements about some mathematical structure, we will want to be able to make assertions about the objects of the structure. These assertions will be the formulas of $\mathcal{L}$.

Definition 1.3.3. If $\mathcal{L}$ is a first-order language, a **formula of** $\mathcal{L}$ is a nonempty finite string ϕ of symbols from $\mathcal{L}$ such that either:

1. $\phi :\equiv = t_1t_2$, where t_1 and t_2 are terms of $\mathcal{L}$, or
2. $\phi :\equiv Rt_1t_2\ldots t_n$, where R is an n-ary relation symbol of $\mathcal{L}$ and t_1, t_2, $\ldots$, t_n are all terms of $\mathcal{L}$, or
3. $\phi :\equiv (\neg\alpha)$, where α is a formula of $\mathcal{L}$, or
4. $\phi :\equiv (\alpha \vee \beta)$, where α and β are formulas of $\mathcal{L}$, or
5. $\phi :\equiv (\forall v)(\alpha)$, where v is a variable and α is a formula of $\mathcal{L}$.

If a formula ψ contains the subformula $(\forall v)(\alpha)$ [meaning that the string of symbols that constitute the formula $(\forall v)(\alpha)$ is a substring of the string of symbols that make up ψ], we will say that the **scope** of the quantifier $\forall$ is α. Any symbol in α will be said to lie within the scope of the quantifier $\forall$. Notice that a formula ψ can have several different occurrences of the symbol $\forall$, and each occurrence of the quantifier will have its own scope. Also notice that one quantifier can lie within the scope of another.

The **atomic formulas of** $\mathcal{L}$ are those formulas that satisfy clause (1) or (2) of Definition 1.3.3.

You have undoubtedly noticed that there are no parentheses or commas in the atomic formulas, and you have probably decided that we will continue

to use both commas and infix notation as seems appropriate. You are correct on both counts. So, instead of writing the official version

$$< SSSSS0SS0$$

in a language containing constant symbol 0, unary function symbol S, and binary relation symbol $<$, we will write

$$SSSSS0 \quad < \quad SS0$$

or (after some preliminary definitions)

$$\overline{5} < \overline{2}.$$

Also notice that we *are* using infix notation for the binary logical connective $\vee$. We hope that this will make your life somewhat easier.

You will be asked in Exercise 8 in Section 1.4.1 to prove that unique readability holds for formulas as well as terms. We will, in our exposition, use different-size parentheses, different shapes of delimiters, and omit parentheses in order to improve readability without (we hope) introducing confusion on your part.

Notice that a term is not a formula! If the terms are the nouns of the language, the formulas will be the statements. Statements can be either true or false. Nouns cannot. Much confusion can be avoided if you keep this simple dictum in mind.

For example, suppose that you are looking at a string of symbols and you notice that the string does not contain either the symbol $=$ or any other relation symbol from the language. Such a string cannot be a formula, as it makes no claim that can be true or false. The string might be a term, it might be nonsense, but it cannot be a formula.

> *Chaff:* We do hope that you have noticed that we are dealing only with the syntax of our language here. We have not mentioned that the symbol $\neg$ will be used for denial, or that $\vee$ will mean "or," or even that $\forall$ means "for every." Don't worry, they will mean what you think they should mean. Similarly, do not worry about the fact that the definition of a formula left out symbols for conjunctions, implications, and biconditionals. We will get to them in good time.

1.3.1 Exercises

1. Suppose that the language $\mathcal{L}$ consists of two constant symbols, $\diamondsuit$ and $\heartsuit$, a unary relation symbol ¥, a binary function symbol $\flat$, and a 3-ary function symbol $\sharp$. Write down at least three distinct terms of the language $\mathcal{L}$. Write down a couple of nonterms that look like they might be terms and explain why they are not terms. Write a couple of formulas and a couple of nonformulas that look like they ought to be formulas.

2. The fact that we write all of our operations on the left is important for unique readability. Suppose, for example, that we wrote our binary operations in the middle (and did not allow the use of parentheses). If our language included the binary function symbol #, then the term

$$u\#v\#w$$

could be interpreted two ways. This can make a difference: Suppose that the operation associated with the function symbol # is "subtract." Find three real numbers u, v, and w such that the two different interpretations of $u\#v\#w$ lead to different answers. Any nonassociative binary function will yield another counterexample to unique readability. Can you think of three such functions?

3. The language of number theory is

$$\mathcal{L}_{NT} \text{ is } \{0, S, +, \cdot, E, <\},$$

where the intended meanings of the symbols are as follows: 0 stands for the number zero, S is the successor function $S(x) = x + 1$, the symbols $+$, $\cdot$, and $<$ mean what you expect, and E stands for exponentiation, so $E(3, 2) = 9$. Assume that $\mathcal{L}_{NT}$-formulas will be interpreted with respect to the nonnegative integers and write an $\mathcal{L}_{NT}$-formula to express the claim that p is a prime number. Can you write the statement of Lagrange's Theorem, which states that every natural number is the sum of four squares?

Write a formula stating that there is no largest prime number. How would we express the Goldbach Conjecture, that every even number greater than two can be expressed as the sum of two primes?

What is the formal statement of the Twin Primes Conjecture, which says that there are infinitely many pairs (x, y) such that x and y are both prime and $y = x + 2$? The Bounded Gap Theorem, proven in 2013, says that there are infinitely many pairs of prime numbers that differ by 70,000,000 or less. Write a formal statement of that theorem.

Use shorthand in your answers to this problem. For example, after you have found the formula which says that p is prime, call the formula $Prime(p)$, and use $Prime(p)$ in your later answers.

4. Suppose that our language has infinitely many constant symbols of the form $', '', ''', \ldots$ and no function or relation symbols other than $=$. Explain why this situation leads to problems by looking at the formula $=''''''$. Where in our definitions do we outlaw this sort of problem?

1.4 Induction

You are familiar, no doubt, with proofs by induction. They are the bane of most mathematics students from their first introduction in high school

through the college years. It is our goal in this section to discuss the proofs by induction that you know so well, put them in a different light, and then generalize that notion of induction to a setting that will allow us to use induction to prove things about terms and formulas rather than just the natural numbers.

Just to remind you of the general form of a proof by induction on the natural numbers, let us state and prove a familiar theorem, assuming for the moment that the set of natural numbers is $\{1, 2, 3, \ldots\}$.

Theorem 1.4.1. *For every natural number n,*

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

Proof. If $n = 1$, simple computation shows that the equality holds. For the inductive case, fix $k \geq 1$ and assume that

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

If we add $k + 1$ to both sides of this equation, we get

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

and simplifying the right-hand side of this equation shows that

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

finishing the inductive step, and the proof. □

As you look at the proof of this theorem, you notice that there is a base case, when $n = 1$, and an inductive case. In the inductive step of the proof, we prove the implication

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

We prove this implication by assuming the antecedent, that the theorem holds for a (fixed, but unknown) number k, and from that assumption proving the consequent, that the theorem holds for the next number, $k+1$. Notice that this is *not* the same as assuming the theorem that we are trying to prove. The theorem is a universal statement—it claims that a certain formula holds for every natural number.

Looking at this from a slightly different angle, what we have done is to construct a set of numbers with a certain property. If we let S stand for the set of numbers for which our theorem holds, in our proof by induction we show the following facts about S:

1. The number 1 is an element of S. We prove this explicitly in the base case of the proof.

2. If the number k is an element of S, then the number $k + 1$ is an element of S. This is the content of the inductive step of the proof.

But now, notice that we know that the collection of natural numbers can be defined as the smallest set such that:

1. The number 1 is a natural number.

2. If k is a natural number, then $k + 1$ is a natural number.

So S, the collection of numbers for which the theorem holds, is identical with the set of natural numbers, thus the theorem holds for every natural number n, as needed. (If you caught the slight lie here, just substitute "superset" where appropriate.)

So what makes a proof by induction work is the fact that the natural numbers can be defined recursively. There is a base case, consisting of the smallest natural number ("1 is a natural number"), and there is a recursive case, showing how to construct bigger natural numbers from smaller ones ("If k is a natural number, then $k + 1$ is a natural number").

Now, let us look at Definition 1.3.3, the definition of a formula. Notice that the five clauses of the definition can be separated into two groups. The first two clauses, the atomic formulas, are explicitly defined: For example, the first case says that anything that is of the form $= t_1 t_2$ is a formula if t_1 and t_2 are terms. These first two clauses form the base case of the definition. The last three clauses are the recursive case, showing how if α and β are formulas, they can be used to build more complex formulas, such as $(\alpha \vee \beta)$ or $(\forall v)(\alpha)$.

Now since the collection of formulas is defined recursively, we can use an inductive-style proof when we want to prove that something is true about *every* formula. The inductive proof will consist of two parts, a base case and an inductive case. In the base case of the proof we will verify that the theorem is true about every atomic formula—about every string that is known to be a formula from the base case of the definition. In the inductive step of the proof, we assume that the theorem is true about simple formulas (α and β), and use that assumption to prove that the theorem holds a more complicated formula ϕ that is generated by a recursive clause of the definition. This method of proof is called *induction on the complexity of the formula,* or *induction on the structure of the formula.*

There are (at least) two ways to think about the word "simple" in the last paragraph. One way in which a formula α might be simpler than a complicated formula ϕ is if α is a subformula of ϕ. The following theorem, although mildly interesting in its own right, is included here mostly so that you can see an example of a proof by induction in this setting:

Theorem 1.4.2. *Suppose that ϕ is a formula in the language $\mathcal{L}$. Then the number of left parentheses occurring in ϕ is equal to the number of right parentheses occurring in ϕ.*

Proof. We will present this proof in a fair bit of detail, in order to emphasize the proof technique. As you become accustomed to proving theorems by induction on complexity, not so much detail is needed.

Base Case. We begin our inductive proof with the base case, as you would expect. Our theorem makes an assertion about all formulas, and the simplest formulas are the atomic formulas. They constitute our base case. Suppose that ϕ is an atomic formula. There are two varieties of atomic formulas: Either ϕ begins with an equals sign followed by two terms, or ϕ begins with a relation symbol followed by several terms. As there are no parentheses in any term (we are using the official definition of term, here), there are no parentheses in ϕ. Thus, there are as many left parentheses as right parentheses in ϕ, and we have established the theorem if ϕ is an atomic formula.

Inductive Case. The inductive step of a proof by induction on complexity of a formula takes the following form: Assume that ϕ is a formula by virtue of clause (3), (4), or (5) of Definition 1.3.3. Also assume that the statement of the theorem is true when applied to the formulas α and β. With those assumptions we will prove that the statement of the theorem is true when applied to the formula ϕ. Thus, as every formula is a formula either by virtue of being an atomic formula or by application of clause (3), (4), or (5) of the definition, we will have shown that the statement of the theorem is true when applied to any formula, which has been our goal.

So, assume that α and β are formulas that contain equal numbers of left and right parentheses. Suppose that there are k left parentheses and k right parentheses in α and l left parentheses and l right parentheses in β.

If ϕ is a formula by virtue of clause (3) of the definition, then $\phi :\equiv (\neg\alpha)$. We observe that there are $k+1$ left parentheses and $k+1$ right parentheses in ϕ, and thus ϕ has an equal number of left and right parentheses, as needed.

If ϕ is a formula because of clause (4), then $\phi :\equiv (\alpha \vee \beta)$, and ϕ contains $k+l+1$ left and right parentheses, an equal number of each type.

Finally, if $\phi :\equiv (\forall v)(\alpha)$, then ϕ contains $k+2$ left parentheses and $k+2$ right parentheses, as needed.

This concludes the possibilities for the inductive case of the proof, so we have established that in every formula, the number of left parentheses is equal to the number of right parentheses. □

A second way in which we might structure a proof by induction on the structure of the formula is to say that α is simpler than ϕ if the number of connectives/quantifiers in α is less than the number in ϕ. In this case one

could argue that the induction argument is really an ordinary induction on the natural numbers. Here is an outline of how such a proof might proceed:

Proof. We argue by induction on the structure of ϕ.

Base Case. Assume ϕ has 0 connectives/quantifiers. This means that ϕ is an atomic formula. {Insert argument establishing the theorem for atomic formulas.}

Inductive Case. Assume that ϕ has $k+1$ connectives/quantifiers. Then either $\phi :\equiv \neg\alpha$, or $\phi :\equiv \alpha \vee \beta$ or $\phi :\equiv (\forall x)\alpha$, and we can assume that the theorem holds for *every* formula that has k or fewer connectives/quantifiers. We now argue that the theorem holds for the formula ϕ. {Insert arguments for the three inductive cases.}

Between the base case and the inductive case we have established that the theorem holds for ϕ no matter how many connectives/quantifiers the formula ϕ contains, so by induction on the structure of ϕ, we have established that the theorem holds for all formulas ϕ.

□

This might be a bit confusing on first glance, but the power of this proof technique will become very evident as you work through the following exercises and when we discuss the semantics of our language.

Notice also that the definition of a term (Definition 1.3.1) is also a recursive definition, so we can use induction on the complexity of a term to prove that a theorem holds for every term.

1.4.1 Exercises

1. Prove, by ordinary induction on the natural numbers, that
$$1^2 + 2^2 + \cdots + n^2 = \frac{n(n+1)(2n+1)}{6}.$$

2. Prove, by induction, that the sum of the interior angles in a convex n-gon is $(n-2)180°$. (A convex n-gon is a polygon with n sides, where the interior angles are all less than 180°.)

3. Prove by induction that if A is a set consisting of n elements, then A has 2^n subsets.

4. Suppose that $\mathcal{L}$ is $\{0, f, g\}$, where 0 is a constant symbol, f is a binary function symbol, and g is a 4-ary function symbol. Use induction on complexity to show that every $\mathcal{L}$-term has an odd number of symbols.

5. If $\mathcal{L}$ is $\{0, <\}$, where 0 is a constant symbol and $<$ is a binary relation symbol, show that the number of symbols in any formula is divisible by 3.

6. If s and t are strings, we say that s is an *initial segment* of t if there is a nonempty string u such that $t :\equiv su$, where su is the string s followed by the string u. For example, KUMQ is an initial segment of KUMQUAT and $+24$ is an initial segment of $+24u - v$. Prove, by induction on the complexity of s, that if s and t are terms, then s is not an initial segment of t. [*Suggestion:* The base case, when s is either a variable or a constant symbol, should be easy. Then suppose that s is an initial segment of t and $s :\equiv ft_1t_2 \dots t_n$, where you know that each t_i is not an initial segment of any other term. Look for a contradiction.]

7. A language is said to satisfy unique readability for terms if, for each term t, t is in exactly one of the following categories:

   (a) Variable

   (b) Constant symbol

   (c) Complex term

   and furthermore, if t is a complex term, then there is a unique function symbol f and a unique sequence of terms $t_1, t_2, \dots, t_n$ such that $t :\equiv ft_1t_2 \dots t_n$. Prove that our languages satisfy unique readability for terms. [*Suggestion:* You mostly have to worry about uniqueness—for example, suppose that t is c, a constant symbol. How do you know that t is not also a complex term? Suppose that t is $ft_1t_2 \dots t_n$. How do you show that the f and the t_i's are unique? You may find Exercise 6 useful.]

8. To say that a language satisfies unique readability for formulas is to say that every formula ϕ is in exactly one of the following categories:

   (a) Equality (if $\phi :\equiv = t_1t_2$)

   (b) Other atomic (if $\phi :\equiv Rt_1t_2 \dots t_n$ for an n-ary relation symbol R)

   (c) Negation

   (d) Disjunction

   (e) Quantified

   Also, it must be that if ϕ is both $= t_1t_2$ and $= t_3t_4$, then t_1 is identical to t_3 and t_2 is identical to t_4, and similarly for other atomic formulas. Furthermore, if (for example) ϕ is a negation $(\neg\alpha)$, then it must be the case that there is not another formula β such that ϕ is also $(\neg\beta)$, and similarly for disjunctions and quantified formulas. Prove that our languages satisfy unique readability for formulas. You will want to look at, and use, Exercise 7. You may have to prove an analog of Exercise 6, in which it may be helpful to think about the parentheses in an initial segment of a formula, in order to prove that no formula is an initial segment of another formula.

9. Take the proof of Theorem 1.4.2 and write it out in the way that you would present it as part of a homework assignment. Thus, you should cut out all of the inessential motivation and present only what is needed to make the proof work.

1.5 Sentences

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

Let us use an example to introduce a language that will be vitally important to us as we work through this book.

Definition 1.5.1. The language $\mathcal{L}_{NT}$ is $\{0, S, +, \cdot, E, <\}$, where 0 is a constant symbol, S is a unary function symbol, $+$, $\cdot$, and E are binary function symbols, and $<$ is a binary relation symbol. This will be referred to as the language of number theory.

> *Chaff:* Although we are not fixing the meanings of these symbols yet, we probably ought to tell you that the standard interpretation of $\mathcal{L}_{NT}$ will use 0, $+$, $\cdot$, and $<$ in the way that you expect. The symbol S will stand for the successor function that maps a number x to the number $x+1$, and E will be used for exponentiation: $E32$ is supposed to be 3^2.

Consider the following two formulas of $\mathcal{L}_{NT}$:

$$\neg(\forall x)[(y < x) \vee (y = x)].$$
$$(\forall x)(\forall y)[(x < y) \vee (x = y) \vee (y < x)].$$

(Did you notice that we have begun using an informal presentation of the formulas?)

The second formula should look familiar. It is nothing more than the familiar trichotomy law of $<$, and you would agree that the second formula is a true statement about the collection of natural numbers, where you are interpreting $<$ in the usual way.

The first formula above is different. It "says" that not every x is greater than or equal to y. The truth of that statement is indeterminate: It depends on what natural number y represents. The formula might be true, or it might be false—it all depends on the value of y. So our goal in this section is to separate the formulas of $\mathcal{L}$ into one of two classes: the sentences (like the second example above) and the nonsentences. To begin this task, we must talk about free variables.

Free variables are the variables upon which the truth value of a formula may depend. The variable y is free in the first formula above. To draw an analogy from calculus, if we look at

$$\int_1^x \frac{1}{t}\, dt,$$

the variable x is free in this expression, as the value of the integral depends on the value of x. The variable t is not free, and in fact it doesn't make any sense to decide on a value for t. The same distinction holds between free and nonfree variables in an $\mathcal{L}$-formula. Let us try to make things a little more precise.

Definition 1.5.2. Suppose that v is a variable and ϕ is a formula. We will say that v **is free in** ϕ if

1. ϕ is atomic and v occurs in (is a symbol in) ϕ, or
2. $\phi :\equiv (\neg\alpha)$ and v is free in α, or
3. $\phi :\equiv (\alpha \vee \beta)$ and v is free in at least one of α or β, or
4. $\phi :\equiv (\forall u)(\alpha)$ and v is not u and v is free in α.

Thus, if we look at the formula

$$\forall v_2 \neg(\forall v_3)(v_1 = S(v_2) \vee v_3 = v_2),$$

the variable v_1 is free whereas the variables v_2 and v_3 are not free. A slightly more complicated example is

$$(\forall v_1 \forall v_2(v_1 + v_2 = 0)) \vee v_1 = S(0).$$

In this formula, v_1 is free whereas v_2 is not free. Especially when a formula is presented informally, you must be careful about the scope of the quantifiers and the placement of parentheses.

We will have occasion to use the informal notation $\forall x \phi(x)$. This will mean that ϕ is a formula and x is among the free variables of ϕ. If we then write $\phi(t)$, where t is an $\mathcal{L}$-term, that will denote the formula obtained by taking ϕ and replacing each occurrence of the variable x with the term t. This will all be defined more formally and more precisely in Definition 1.8.2.

Definition 1.5.3. A **sentence** in a language $\mathcal{L}$ is a formula of $\mathcal{L}$ that contains no free variables.

For example, if a language contained the constant symbols 0, 1, and 2 and the binary function symbol $+$, then the following are sentences: $1+1 = 2$ and $(\forall x)(x + 1 = x)$. You are probably convinced that the first of these is true and the second of these is false. In the next two sections we will see that you might be correct. But then again, you might not be.

1.5.1 Exercises

1. For each of the following, find the free variables, if any, and decide if the given formula is a sentence. The language includes a binary function symbol $+$, a binary relation symbol $<$, and constant symbols 0 and 2.

   (a) $(\forall x)(\forall y)(x + y = 2)$

   (b) $(x + y < x) \vee (\forall z)(z < 0)$

   (c) $((\forall y)(y < x)) \vee ((\forall x)(x < y))$

2. Explain precisely, using the definition of a free variable, how you know that the variable v_2 is free in the formula

$$(\forall v_1)(\neg(\forall v_5)(v_2 = v_1 + v_5)).$$

3. In mathematics, we often see statements such as $\sin^2 x + \cos^2 x = 1$. Notice that this is not a sentence, as the variable x is free. But we all agree that this statement is true, given the usual interpretations of the symbols. How can we square this with the claim that *sentences* are the formulas that can be either true or false?

4. If we look at the first of our example formulas in this section,

$$\neg(\forall x)[(y < x) \vee (y = x)],$$

   and we interpret the variables as ranging over the natural numbers, you will probably agree that the formula is false if y represents the natural number 0 and true if y represents any other number. (If you aren't happy with 0 being a natural number, then use 1.) On the other hand, if we interpret the variables as ranging over the integers, what can we say about the truth or falsehood of this formula? Can you think of an interpretation for the symbols that would make sense if we try to apply this formula to the collection of complex numbers?

5. A variable may occur several times in a given formula. For example, the variable v_1 occurs four times in the formula

$$(\forall v_1)\big[(v_1 = v_3) \vee (v_1 = Sv_2) \vee (0 + v_{17} < v_1 - S0)\big].$$

   What should it mean for an *occurrence* of a variable to be free? Write a definition that begins: The nth occurrence of a variable v in a formula ϕ is said to be free if .... An occurrence of v in ϕ that is not free is said to be **bound**. Give an example of a formula in a suitable language that contains both free and bound occurrences of a variable v.

6. Look at the formula

$$\big[(\forall y)(x = y)\big] \vee \big[(\forall x)(x < 0)\big].$$

If we denote this formula by $\phi(x)$ and t is the term $S0$, find $\phi(t)$. [*Suggestion:* The trick here is to see that there is a bit of a lie in the discussion of $\phi(t)$ in the text. Having completed Exercise 5, we can now say that we only replace the free occurrences of the variable x when we move from $\phi(x)$ to $\phi(t)$.]

1.6 Structures

Let us, by way of example, return to the language $\mathcal{L}_{NT}$ of number theory. Recall that $\mathcal{L}_{NT}$ is $\{0, S, +, \cdot, E, <\}$, where 0 is a constant symbol, S is a unary function symbol, $+$, $\cdot$, and E are binary function symbols, and $<$ is a binary relation symbol. We now want to discuss the possible mathematical structures in which we can interpret these symbols, and thus the formulas and sentences of $\mathcal{L}_{NT}$.

"But wait!" cries the incredulous reader. "You just said that this is the language of number theory, so certainly we already know what each of those symbols means."

It is certainly the case that you know *an* interpretation for these symbols. The point of this section is that there are *many* different possible interpretations for these symbols, and we want to be able to specify which of those interpretations we have in mind at any particular moment.

Probably the interpretation you had in mind (what we will call the standard model for number theory) works with the set of natural numbers $\{0, 1, 2, 3, \dots\}$. The symbol 0 stands for the number 0.

> *Chaff:* Carefully, now! The symbol 0 is the mark on the paper, the numeral. The number 0 is the thing that the numeral 0 represents. The numeral is something that you can see. The number is something that you cannot see.

The symbol S is a unary function symbol, and the function for which that symbol stands is the successor function that maps a number to the next larger natural number. The symbols $+, \cdot$, and E represent the functions of addition, multiplication, and exponentiation, and the symbol $<$ will be used for the "less than" relation.

But that is only one of the ways that we might choose to interpret those symbols. Another way to interpret all of those symbols would be to work with the numbers 0 and 1, interpreting the symbol 0 as the number 0, S as the function that maps 0 to 1 and 1 to 0, $+$ as addition mod 2, $\cdot$ as multiplication mod 2, and (just for variety) E as the function with constant value 1. The symbol $<$ can still stand for the relation "less than."

Or, if we were in a slightly more bizarre mood, we could work in a universe consisting of Beethoven, Picasso, and Ernie Banks, interpreting the symbol 0 as Picasso, S as the identity function, $<$ as equality, and each

of the binary function symbols as the constant function with output Ernie Banks.

The point is that there is nothing sacred about one mathematical structure as opposed to another. Without determining the structure under consideration, without deciding how we wish to interpret the symbols of the language, we have no way of talking about the truth or falsity of a sentence as trivial as

$$(\forall v_1)(v_1 < S(v_1)).$$

Definition 1.6.1. Fix a language $\mathcal{L}$. An **$\mathcal{L}$-structure** $\mathfrak{A}$ is a nonempty set A, called the **universe of** $\mathfrak{A}$, together with:

1. For each constant symbol c of $\mathcal{L}$, an element $c^{\mathfrak{A}}$ of A,
2. For each n-ary function symbol f of $\mathcal{L}$, a function $f^{\mathfrak{A}} : A^n \to A$, and
3. For each n-ary relation symbol R of $\mathcal{L}$, an n-ary relation $R^{\mathfrak{A}}$ on A (i.e., a subset of A^n).

Notice that the domain of the function $f^{\mathfrak{A}}$ is the set A^n, so $f^{\mathfrak{A}}$ is defined for all elements of A^n. Later in the text we will have occasion to discuss partial functions, those whose domain is a proper subset of A^n, but for now our functions are total functions, defined on all of the advertised domain.

> *Chaff:* The letter $\mathfrak{A}$ is a German Fraktur capital A. We will also have occasion to use $\mathfrak{A}$'s friends, $\mathfrak{B}$ and $\mathfrak{C}$. $\mathfrak{N}$ will be used for a particular structure involving the natural numbers. The use of this typeface is traditional (which means this is the way we learned it). For your handwritten work, probably using capital script letters will be the best.

Often, we will write a structure as an ordered k-tuple, like this:

$$\mathfrak{A} = (A, c_1^{\mathfrak{A}}, c_2^{\mathfrak{A}}, f_1^{\mathfrak{A}}, R_1^{\mathfrak{A}}, R_2^{\mathfrak{A}}).$$

As you can see, the notation is starting to get out of hand once again, and we will not hesitate to simplify and abbreviate when we believe that we can do so without confusion. So, when we are working in $\mathcal{L}_{NT}$, we will often talk about the standard structure

$$\mathfrak{N} = (\mathbb{N}, 0, S, +, \cdot, E, <),$$

where the constants, functions, and relations do not get the superscripts they deserve, and the authors trust that you will interpret $\mathbb{N}$ as the collection $\{0, 1, 2, \ldots\}$ of natural numbers, the symbol 0 to stand for the number zero, $+$ to stand for addition, S to stand for the successor function, and so on. By the way, if you are not used to thinking of 0 as a natural number, do not panic. Set theorists see 0 as the most natural of objects, so we tend to include it in $\mathbb{N}$ without thinking about it.

x	$S^{\mathfrak{A}}(x)$
Oberon	Oberon
Titania	Bottom
Puck	Titania
Bottom	Titania

$+^{\mathfrak{A}}$	Oberon	Titania	Puck	Bottom
Oberon	Puck	Puck	Puck	Titania
Titania	Puck	Bottom	Oberon	Titania
Puck	Bottom	Titania	Bottom	Titania
Bottom	Bottom	Bottom	Bottom	Oberon

$\cdot^{\mathfrak{A}}$	Oberon	Titania	Puck	Bottom
Oberon	Oberon	Titania	Puck	Bottom
Titania	Titania	Bottom	Oberon	Titania
Puck	Bottom	Bottom	Oberon	Oberon
Bottom	Titania	Oberon	Puck	Puck

$E^{\mathfrak{A}}$	Oberon	Titania	Puck	Bottom
Oberon	Puck	Puck	Oberon	Oberon
Titania	Titania	Titania	Titania	Titania
Puck	Titania	Bottom	Oberon	Puck
Bottom	Bottom	Puck	Titania	Puck

$<^{\mathfrak{A}}$	Oberon	Titania	Puck	Bottom
Oberon	Yes	No	Yes	Yes
Titania	No	No	Yes	No
Puck	Yes	Yes	Yes	Yes
Bottom	No	No	Yes	No

Table 1.1: A Midsummer Night's Structure

Example 1.6.2. The structure $\mathfrak{N}$ that we have just introduced is called the standard $\mathcal{L}_{NT}$-structure. To emphasize that there are other perfectly good $\mathcal{L}_{NT}$-structures, let us construct a different $\mathcal{L}_{NT}$-structure $\mathfrak{A}$ with exactly four elements. The elements of A will be Oberon, Titania, Puck, and Bottom. The constant $0^{\mathfrak{A}}$ will be Bottom. Now we have to construct the functions and relations for our structure. As everything is unary or binary, setting forth tables (as in Table 1.1) seems a reasonable way to proceed. So you can see that in this structure $\mathfrak{A}$ that Titania + Puck = Oberon, while Puck + Titania = Titania. You can also see that 0 (also known as Bottom) is not the additive identity in this structure, and that $<$ is a very strange ordering.

Now the particular functions and relation that we chose were just the

functions and relations that jumped into Chris's fingers as he typed up this example, but any such functions would have worked perfectly well to define an $\mathcal{L}_{NT}$-structure. It may well be worth your while to figure out if this $\mathcal{L}_{NT}$-sentence is true (whatever that means) in $\mathfrak{A}$: $SS0{+}SS0 < SSSSS0E0{+}S0$.

Example 1.6.3. We work in a language with one constant symbol, $\pounds$, and one unary function symbol, X. So, to define a model $\mathfrak{A}$, all we need to do is specify a universe, an element of the universe, and a function $X^{\mathfrak{A}}$. Suppose that we let the universe be the collection of all finite strings of 0 or more capital letters from the Roman alphabet. So A includes such strings as: BABY, LOGICISBETTERTHANSIX, ε (the empty string), and DLKFDFAHADS. The constant symbol $\pounds$ will be interpreted as the string POTITION, and the function $X^{\mathfrak{A}}$ is the function that adds an X to the beginning of a string. So $X^{\mathfrak{A}}(\text{YLOPHONE}) = \text{XYLOPHONE}$. Convince yourself that this is a valid, if somewhat odd, $\mathcal{L}$-structure.

To try to be clear about things, notice that we have X, the function symbol, which is an element of the language $\mathcal{L}$. Then there is X, the string of exactly one capital letter of the Roman alphabet, which is one of the elements of the universe. (Did you notice the change in typeface without our pointing it out? You may have a future in publishing!)

Let us look at one of the terms of the language: $X\pounds$. In our particular $\mathcal{L}$-structure $\mathfrak{A}$ we will interpret this as

$$X^{\mathfrak{A}}(\pounds^{\mathfrak{A}}) = X^{\mathfrak{A}}(\text{POTITION}) = \text{XPOTITION}.$$

In a different structure, $\mathfrak{B}$, it is entirely possible that the interpretation of the term $X\pounds$ will be HUNNY or AARDVARK or $3\pi/17$. Without knowing the structure, without knowing how to interpret the symbols of the language, we cannot begin to know what object is referred to by a term.

> *Chaff:* All of this stuff about interpreting terms in a structure will be made formal in the next section, so don't panic if it doesn't all make sense right now.

What makes this example confusing, as well as important, is that the function symbol is part of the structure for the language and (modulo a superscript and a change in typeface) the function acts on the elements of the structure in the same way that the function symbol is used in creating $\mathcal{L}$-formulas.

Example 1.6.4. Now, let $\mathcal{L}$ be $\{0, f, g, R\}$, where 0 is a constant symbol, f is a unary function symbol, g is a binary function symbol, and R is a 3-ary relation symbol. We define an $\mathcal{L}$-structure $\mathfrak{B}$ as follows: B, the universe, is the set of all variable-free $\mathcal{L}$-terms. The constant $0^{\mathfrak{B}}$ is the term 0. The functions $f^{\mathfrak{B}}$ and $g^{\mathfrak{B}}$ are defined as in Example 1.6.3, so if t and s are elements of B (i.e., variable-free terms), then $f^{\mathfrak{B}}(t)$ is ft and $g^{\mathfrak{B}}(t, s)$ is gts.

Let us look at this in a little more detail. Consider 0, the constant symbol, which is an element of $\mathcal{L}$. Since 0 is a constant symbol, it is a term, so 0 is an element of B, the universe of our structure $\mathfrak{B}$. (Alas, there is no change in typeface to help us out this time.) If we want to see what element of the universe is referred to by the constant symbol 0, we see that $0^{\mathfrak{B}} = 0$, so the term 0 refers to the element of the universe 0.

If we look at another term of the language, say, $f0$, and we try to find the element of the universe that is denoted by this term, we find that it is

$$f^{\mathfrak{B}}(0^{\mathfrak{B}}) = f^{\mathfrak{B}}(0) = f0.$$

So the term $f0$ denotes an element of the universe, and that element of the universe is $\ldots f0$. This is pretty confusing, but all that is going on is that the elements of the universe *are* the syntactic objects of the language.

This sort of structure is called a *Henkin structure*, after Leon Henkin, who introduced them in his PhD dissertation in 1949. These structures will be crucial in our proof of the Completeness Theorem in Chapter 3. The proof of that theorem will involve the construction of a particular mathematical structure, and the structure that we will build will be a Henkin structure.

To finish building our structure $\mathfrak{B}$, we have to define a relation $R^{\mathfrak{B}}$. As R is a 3-ary relation symbol, $R^{\mathfrak{B}}$ is a subset of B^3. We will arbitrarily define

$$R^{\mathfrak{B}} = \{(r, s, t) \in B^3 \mid \text{the number of function symbols in } r \text{ is even}\}.$$

This finishes defining the structure $\mathfrak{B}$. The definition of $R^{\mathfrak{B}}$ given is entirely arbitrary. We invite you to come up with a more interesting or more humorous definition on your own.

1.6.1 Exercises

1. Consider the structure constructed in Example 1.6.2. Find the value of each of the following: $0 + 0$, $0E0$, $S0 \cdot SS0$. Do you think $0 < 0$ in this structure?

2. Suppose that $\mathcal{L}$ is the language $\{0, +, <\}$. Let's work together to describe an $\mathcal{L}$-structure $\mathfrak{A}$. Let the universe A be the set consisting of all of the natural numbers together with Ingrid Bergman and Humphrey Bogart. You decide on the interpretations of the symbols. What is the value of $5 +$ Ingrid? Is Bogie < 0? (Suggested solution on page 286.)

3. Here is a language consisting of one constant symbol, one 3-ary function symbol, and one binary relation symbol: $\mathcal{L}$ is $\{\flat, \sharp, \natural\}$. Describe an $\mathcal{L}$-model that has as its universe $\mathbb{R}$, the set of real numbers. Describe another $\mathcal{L}$-model that has a finite universe.

4. Write a short paragraph explaining the difference between a language and a structure for a language.

5. Suppose that $\mathfrak{A}$ and $\mathfrak{B}$ are two $\mathcal{L}$-structures. We will say that $\mathfrak{A}$ and $\mathfrak{B}$ are **isomorphic** and write $\mathfrak{A} \cong \mathfrak{B}$ if there is a bijection $i : A \to B$ such that for each constant symbol c of $\mathcal{L}$, $i(c^{\mathfrak{A}}) = c^{\mathfrak{B}}$, for each n-ary function symbol f and for each $a_1, \ldots, a_n \in A$, $i(f^{\mathfrak{A}}(a_1, \ldots, a_n)) = f^{\mathfrak{B}}(i(a_1), \ldots, i(a_n))$, and for each n-ary relation symbol R in $\mathcal{L}$, $(a_1, \ldots, a_n) \in R^{\mathfrak{A}}$ if and only if $(i(a_1), \ldots, i(a_n)) \in R^{\mathfrak{B}}$. The function i is called an **isomorphism**.

   (a) Show that $\cong$ is an equivalence relation. [*Suggestion:* This means that you must show that the relation $\cong$ is reflexive, symmetric, and transitive. To show that $\cong$ is reflexive, you must show that for any structure $\mathfrak{A}$, $\mathfrak{A} \cong \mathfrak{A}$, which means that you must find an isomorphism, a function, mapping A to A that satisfies the conditions above. So the first line of your proof should be, "Consider this function, with domain A and codomain A: $i(x) =$ something brilliant." Then show that your function i is an isomorphism. Then show, if $\mathfrak{A} \cong \mathfrak{B}$, then $\mathfrak{B} \cong \mathfrak{A}$. Then tackle transitivity. In each case, you must define a particular function and show that your function is an isomorphism.]

   (b) Find a new structure that is isomorphic to the structure given in Example 1.6.2. Prove that the structures are isomorphic.

   (c) Find two different structures for a particular language and prove that they are not isomorphic.

   (d) Find two different structures for a particular language such that the structures have the same number of elements in their universes but they are still not isomorphic. Prove they are not isomorphic.

6. Take the language of Example 1.6.4 and let C be the set of all $\mathcal{L}$-terms. Create an $\mathcal{L}$-structure $\mathfrak{C}$ by using this universe in such a way that the interpretation of a term t is *not* equal to t.

7. If we take the language $\mathcal{L}_{NT}$, we can create a Henkin structure for that language in the same way as in Example 1.6.4. Do so. Consider the $\mathcal{L}_{NT}$-formula $S0 + S0 = SS0$. Is this formula "true" (whatever that means) in your structure? Justify your answer.

1.7 Truth in a Structure

It is at last time to tie together the syntax and the semantics. We have some formal rules about what constitutes a language, and we can identify the terms, formulas, and sentences of a language. We can also identify

$\mathcal{L}$-structures for a given language $\mathcal{L}$. In this section we will decide what it means to say that an $\mathcal{L}$-formula ϕ is *true* in an $\mathcal{L}$-structure $\mathfrak{A}$.

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

Definition 1.7.1. If $\mathfrak{A}$ is an $\mathcal{L}$-structure, a **variable assignment function into** $\mathfrak{A}$ is a function s that assigns to each variable an element of the universe A. So a variable assignment function into $\mathfrak{A}$ is any function with domain *Vars* and codomain A.

Variable assignment functions need not be injective or bijective. For example, if we work with $\mathcal{L}_{NT}$ and the standard structure $\mathfrak{N}$, then the function s defined by $s(v_i) = i$ is a variable assignment function, as is the function s' defined by

$$s'(v_i) = \text{the smallest prime number that does not divide } i.$$

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

Definition 1.7.2. If s is a variable assignment function into $\mathfrak{A}$ and x is a variable and $a \in A$, then $s[x|a]$ is the variable assignment function into $\mathfrak{A}$ defined as follows:

$$s[x|a](v) = \begin{cases} s(v) & \text{if } v \text{ is a variable other than } x \\ a & \text{if } v \text{ is the variable } x. \end{cases}$$

We call the function $s[x|a]$ an **x-modification of the assignment function** s.

So an x-modification of s is just like s, except that the variable x is assigned to a particular element of the universe.

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

Definition 1.7.3. Suppose that $\mathfrak{A}$ is an $\mathcal{L}$-structure and s is a variable assignment function into $\mathfrak{A}$. The function $\overline{s}$, called the **term assignment function generated by** s, is the function with domain consisting of the set of $\mathcal{L}$-terms and codomain A defined recursively as follows:

1. If t is a variable, $\overline{s}(t) = s(t)$.
2. If t is a constant symbol c, then $\overline{s}(t) = c^{\mathfrak{A}}$.
3. If $t :\equiv ft_1t_2 \dots t_n$, then $\overline{s}(t) = f^{\mathfrak{A}}(\overline{s}(t_1), \overline{s}(t_2), \dots, \overline{s}(t_n))$.

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

Definition 1.7.4. Suppose that $\mathfrak{A}$ is an $\mathcal{L}$-structure, ϕ is an $\mathcal{L}$-formula, and $s : \mathit{Vars} \to A$ is an assignment function. We will say that $\mathfrak{A}$ **satisfies** ϕ **with assignment** s, and write $\mathfrak{A} \models \phi[s]$, in the following circumstances:

1. If $\phi :\equiv = t_1 t_2$ and $\overline{s}(t_1)$ is the same element of the universe A as $\overline{s}(t_2)$, or
2. If $\phi :\equiv Rt_1 t_2 \dots t_n$ and $(\overline{s}(t_1), \overline{s}(t_2), \dots, \overline{s}(t_n)) \in R^{\mathfrak{A}}$, or
3. If $\phi :\equiv (\neg\alpha)$ and $\mathfrak{A} \not\models \alpha[s]$, (where $\not\models$ means "does not satisfy") or
4. If $\phi :\equiv (\alpha \vee \beta)$ and $\mathfrak{A} \models \alpha[s]$, or $\mathfrak{A} \models \beta[s]$ (or both), or
5. If $\phi :\equiv (\forall x)(\alpha)$ and, for each element a of A, $\mathfrak{A} \models \alpha[s(x|a)]$.

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

> *Chaff:* Notice that the symbol $\models$ is *not* part of the language $\mathcal{L}$. Rather, $\models$ is a metalinguistic symbol that we use to talk about formulas in the language and structures for the language.

> *Chaff:* Also notice that we have at last tied together the syntax and the semantics of our language! The definition above is the place where we formally put the meanings on the symbols that we will use, so that $\vee$ means "or" and $\forall$ means "for all."

Example 1.7.5. Let us work with the empty language, so $\mathcal{L}$ has no constant symbols, no function symbols, and no relation symbols. So an $\mathcal{L}$-structure is simply a nonempty set, and let us consider the $\mathcal{L}$-structure $\mathfrak{A}$, where $A = \{\text{Humphrey}, \text{Ingrid}\}$. Consider the formula $x = y$ and the assignment function s, where $s(x)$ is Humphrey and $s(y)$ is also Humphrey. If we ask whether $\mathfrak{A} \models x = y[s]$, we have to check whether $\overline{s}(x)$ is the same element of A as $\overline{s}(y)$. Since the two objects are identical, the formula is true.

To emphasize this, the formula $x = y$ can be true in some universes with some assignment functions. Although the variables x and y are distinct, the truth or falsity of the formula depends *not* on the variables (which are not equal) but rather, on which elements of the structure the variables denote, the *values* of the variables (which are equal for this example). Of course, there are other assignment functions and other structures that make our formula false. We are sure you can think of some.

To talk about the truth or falsity of a *sentence* in a structure, we will take our definition of satisfaction relative to an assignment function and prove that for sentences, the choice of the assignment function is inconsequential. Then we will say that a sentence σ is true in a structure $\mathfrak{A}$ if and only if $\mathfrak{A} \models \sigma[s]$ for any (and therefore all) variable assignment functions s.

> *Chaff:* The next couple of proofs are proofs by induction on the complexity of terms or formulas. You may want to reread the proof of Theorem 1.4.2 on page 16 if you find these difficult.

Lemma 1.7.6. *Suppose that s_1 and s_2 are variable assignment functions into a structure $\mathfrak{A}$ such that $s_1(v) = s_2(v)$ for every variable v in the term t. Then $\overline{s_1}(t) = \overline{s_2}(t)$.*

Proof. We use induction on the complexity of the term t. If t is either a variable or a constant symbol, the result is immediate. If $t :\equiv ft_1t_2 \dots t_n$, then as $\overline{s_1}(t_i) = \overline{s_2}(t_i)$ for $1 \leq i \leq n$ by the inductive hypothesis, the definition of $\overline{s_1}(t)$ and the definition of $\overline{s_2}(t)$ are identical, and thus $\overline{s_1}(t) = \overline{s_2}(t)$. □

Proposition 1.7.7. *Suppose that s_1 and s_2 are variable assignment functions into a structure $\mathfrak{A}$ such that $s_1(v) = s_2(v)$ for every free variable v in the formula ϕ. Then $\mathfrak{A} \models \phi[s_1]$ if and only if $\mathfrak{A} \models \phi[s_2]$.*

Proof. We use induction on the complexity of ϕ. If $\phi :\equiv = t_1t_2$, then the free variables of ϕ are exactly the variables that occur in ϕ. Thus Lemma 1.7.6 tells us that $\overline{s_1}(t_1) = \overline{s_2}(t_1)$ and $\overline{s_1}(t_2) = \overline{s_2}(t_2)$, meaning that they are the same element of the universe A, so $\mathfrak{A} \models (= t_1t_2)[s_1]$ if and only if $\mathfrak{A} \models (= t_1t_2)[s_2]$, as needed.

The other base case, if $\phi :\equiv Rt_1t_2 \dots t_n$, is similar and is left as part of Exercise 6.

To begin the first inductive clause, if $\phi :\equiv \neg\alpha$, notice that the free variables of ϕ are exactly the free variables of α, so s_1 and s_2 agree on the free variables of α. By the inductive hypothesis, $\mathfrak{A} \models \alpha[s_1]$ if and only if $\mathfrak{A} \models \alpha[s_2]$, and thus (by the definition of satisfaction), $\mathfrak{A} \models \phi[s_1]$ if and only if $\mathfrak{A} \models \phi[s_2]$. The second inductive clause, if $\phi :\equiv \alpha \vee \beta$, is another part of Exercise 6.

If $\phi :\equiv (\forall x)(\alpha)$, we first note that the only variable that might be free in α that is not free in ϕ is x. Thus, if $a \in A$, the assignment functions $s_1[x|a]$ and $s_2[x|a]$ agree on all of the free variables of α. Therefore, by inductive hypothesis, for each $a \in A$, $\mathfrak{A} \models \alpha[s_1[x|a]]$ if and only if $\mathfrak{A} \models \alpha[s_2[x|a]]$. So, by Definition 1.7.4, $\mathfrak{A} \models \phi[s_1]$ if and only if $\mathfrak{A} \models \phi[s_2]$. This finishes the last inductive clause, and our proof. □

Corollary 1.7.8. *If σ is a sentence in the language $\mathcal{L}$ and $\mathfrak{A}$ is an $\mathcal{L}$-structure, either $\mathfrak{A} \models \sigma[s]$ for all assignment functions s, or $\mathfrak{A} \models \sigma[s]$ for no assignment function s.*

Proof. There *are* no free variables in σ, so if s_1 and s_2 are two assignment functions, they agree on all of the free variables of σ, there just aren't all that many of them. So by Proposition 1.7.7, $\mathfrak{A} \models \sigma[s_1]$ if and only if $\mathfrak{A} \models \sigma[s_2]$, as needed. □

Definition 1.7.9. If ϕ is a formula in the language $\mathcal{L}$ and $\mathfrak{A}$ is an $\mathcal{L}$-structure, we say that $\mathfrak{A}$ is a **model** of ϕ, and write $\mathfrak{A} \models \phi$, if and only if $\mathfrak{A} \models \phi[s]$ for every assignment function s. If Φ is a set of $\mathcal{L}$-formulas, we will say that $\mathfrak{A}$ models Φ, and write $\mathfrak{A} \models \Phi$, if and only if $\mathfrak{A} \models \phi$ for each $\phi \in \Phi$.

Notice that if σ is a *sentence*, then $\mathfrak{A} \models \sigma$ if and only if $\mathfrak{A} \models \sigma[s]$ for *any* assignment function s. In this case we will say that the sentence σ is **true in** $\mathfrak{A}$.

Example 1.7.10. Let's work in $\mathcal{L}_{NT}$, and let

$$\mathfrak{N} = (\mathbb{N}, 0, S, +, \cdot, E, <)$$

be the standard structure. Let s be the variable assignment function that assigns v_i to the number $2i$. Now let the formula $\phi(v_1)$ be $v_1 + v_1 = SSSS0$.

To show that $\mathfrak{N} \models \phi[s]$, notice that

$$\begin{array}{lll} \overline{s}(v_1 + v_1) & \text{is} & +^{\mathfrak{N}}\big(\overline{s}(v_1), \overline{s}(v_1)\big) \\ & \text{is} & +^{\mathfrak{N}}(2, 2) \\ & \text{is} & 4 \end{array}$$

while

$$\begin{array}{lll} \overline{s}(SSSS0) & \text{is} & S^{\mathfrak{N}}(S^{\mathfrak{N}}(S^{\mathfrak{N}}(S^{\mathfrak{N}}(0^{\mathfrak{N}})))) \\ & \text{is} & 4. \end{array}$$

Now, in the same setting, consider σ, the sentence

$$(\forall v_1)\neg(\forall v_2)\neg(v_1 = v_2 + v_2),$$

which states that everything is even. [That is hard to see unless you know to look for that $\neg(\forall v_2)\neg$ and to read it as $(\exists v_2)$. See the last couple of paragraphs of this section.] You know that σ is false in the standard structure, but to show how the formal argument goes, let s be any variable assignment function and notice that

$$\begin{array}{lll} \mathfrak{N} \models \sigma[s] & \text{iff} & \text{For every } a \in \mathbb{N},\ \mathfrak{N} \models \neg(\forall v_2)\neg(v_1 = v_2 + v_2)s[v_1|a] \\ & \text{iff} & \text{For every } a \in \mathbb{N},\ \mathfrak{N} \not\models (\forall v_2)\neg(v_1 = v_2 + v_2)s[v_1|a] \\ & \text{iff} & \text{For every } a \in \mathbb{N}, \text{ there is a } b \in \mathbb{N}, \\ & & \mathfrak{N} \models v_1 = v_2 + v_2 s[v_1|a][v_2|b]. \end{array}$$

Now, if we consider the case when a is the number 3, it is perfectly clear that there is no such b, so we have shown $\mathfrak{N} \not\models \sigma[s]$. Then, by Definition 1.7.9, we see that the sentence σ is false in the standard structure. As you well knew.

When you were introduced to symbolic logic, you were probably told that there were five connectives. In the mathematics that you have learned recently, you have been using two quantifiers. We hope you have noticed that we have not used all of those symbols in this book, but it is now time to make those symbols available. Rather than adding the symbols to our language, however, we will introduce them as abbreviations. This will help us to keep our proofs slightly less complex (as our inductive proofs will have fewer cases) but will still allow us to use the more familiar symbols, at least as shorthand.

Thus, let us agree to use the following abbreviations in constructing $\mathcal{L}$-formulas: We will write $(\alpha \wedge \beta)$ instead of $(\neg((\neg\alpha) \vee (\neg\beta)))$, $(\alpha \to \beta)$ instead of $((\neg\alpha) \vee \beta)$, and $(\alpha \leftrightarrow \beta)$ instead of $((\alpha \to \beta) \wedge (\beta \to \alpha))$. We will also introduce our missing existential quantifier as an abbreviation, writing $(\exists x)(\alpha)$ instead of $(\neg(\forall x)(\neg\alpha))$. It is an easy exercise to check that the introduced connectives $\wedge$, $\to$, and $\leftrightarrow$ behave as you would expect them to. Thus $\mathfrak{A} \models (\alpha \wedge \beta)[s]$ if and only if both $\mathfrak{A} \models \alpha[s]$ and $\mathfrak{A} \models \beta[s]$. The existential quantifier is only slightly more difficult. See Exercise 7.

1.7.1 Exercises

1. We suggested after Definition 1.5.3 that the truth or falsity of the sentences $1 + 1 = 2$ and $(\forall x)(x + 1 = x)$ might not be automatic. Find a structure for the language discussed there that makes the sentence $1 + 1 = 2$ true. Find another structure where $1 + 1 = 2$ is false. Prove your assertions. Then show that you can find a structure where $(\forall x)(x + 1 = x)$ is true, and another structure where it is false.

2. Let the language $\mathcal{L}$ be $\{S, <\}$, where S is a unary function symbol and $<$ is a binary relation symbol. Let ϕ be the formula $(\forall x)(\exists y)(Sx < y)$.

   (a) Find an $\mathcal{L}$-structure $\mathfrak{A}$ such that $\mathfrak{A} \models \phi$.

   (b) Find an $\mathcal{L}$-structure $\mathfrak{B}$ such that $\mathfrak{B} \models (\neg\phi)$.

   (c) Prove that your answer to part (a) or part (b) is correct.

   (d) Write an $\mathcal{L}$-sentence that is true in a structure $\mathfrak{A}$ if and only if the universe A of $\mathfrak{A}$ consists of exactly two elements.

3. Consider the language and structure of Example 1.6.4. Write two nontrivial sentences in the language, one of which is true in the structure and one of which (not the denial of the first) is false in the structure. Justify your assertions.

4. Consider the sentence σ: $(\forall x)(\exists y)\big[x < y \to x + 1 \neq y\big]$. Find two structures for a suitable language, one of which makes σ true, and the other of which makes σ false.

5. One more bit of shorthand. Assume that the language $\mathcal{L}$ contains the binary relation symbol $\in$, which you are intending to use to mean the elementhood relation (so $p \in q$ will mean that p is an element of q). Often, it is the case that you want to claim that $\phi(x)$ is true for every element of a set b. Of course, to do this you could write

$$(\forall x)\big[(x \in b) \to \phi(x)\big].$$

   We will abbreviate this formula as

$$(\forall x \in b)(\phi(x)).$$

   Similarly, $(\exists x \in b)(\phi(x))$ will be an abbreviation for the formula $(\exists x)\big[(x \in b) \wedge \phi(x)\big]$. Notice that this formula has a conjunction where the previous formula had an implication! We do that just to see if you are paying attention. (Well, if you think about what the abbreviations are supposed to mean, you'll see that the change is necessary. We'll have to do something else just to see if you're paying attention.)

   Now suppose that $\mathfrak{A}$ is a structure for the language of set theory. So $\mathcal{L}$ has only this one binary relation symbol, $\in$, which is interpreted as the elementhood relation. Suppose, in addition, that

$$A = \{u, v, w, \{u\}, \{u, v\}, \{u, v, w\}\}.$$

   In particular, notice that there is no element x of A such that $x \in x$. Consider the sentence

$$(\forall y \in y)(\exists x \in x)(x = y).$$

   Is this sentence true or false in $\mathfrak{A}$?

6. Fill in the details to complete the proof of Proposition 1.7.7.

7. Show that $\mathfrak{A} \models (\exists x)(\alpha)[s]$ if and only if there is an element $a \in A$ such that $\mathfrak{A} \models \alpha[s[x|a]]$.

1.8 Substitutions and Substitutability

Suppose you knew that the sentence $\forall x \phi(x)$ was true in a particular structure $\mathfrak{A}$. Then, if c is a constant symbol in the language, you would certainly expect $\phi(c)$ to be true in $\mathfrak{A}$ as well. What we have done is substitute the constant symbol c for the variable x. This seems perfectly reasonable, although there are times when you do have to be careful.

Suppose that $\mathfrak{A} \models \forall x \exists y \neg(x = y)$. This sentence is, in fact, true in any structure $\mathfrak{A}$ such that A has at least two elements. If we then proceed to replace the variable x by the variable u, we get the statement $\exists y \neg(u = y)$, which will still be true in $\mathfrak{A}$, no matter what value we give to the variable u. If, however, we take our original formula and replace x by y, then we find ourselves looking at $\exists y \neg(y = y)$, which will be false in any structure. So by a poor choice of substituting variable, we have changed the truth value of our formula. The rules of substitutability that we will discuss in this section are designed to help us avoid this problem, the problem of attempting to substitute a term inside a quantifier that binds a variable involved in the term.

We begin by defining exactly what we mean when we substitute a term t for a variable x in either a term u or a formula ϕ.

Definition 1.8.1. Suppose that u is a term, x is a variable, and t is a term. We define the term u_t^x (read "u with x replaced by t") as follows:

1. If u is a variable not equal to x, then u_t^x is u.
2. If u is x, then u_t^x is t.
3. If u is a constant symbol, then u_t^x is u.
4. If $u :\equiv f u_1 u_2 \dots u_n$, where f is an n-ary function symbol and the u_i are terms, then
$$u_t^x \text{ is } f(u_1)_t^x (u_2)_t^x \dots (u_n)_t^x.$$

Chaff: In the fourth clause of the definition above and in the first two clauses of the next definition, the parentheses are not really there. However, we believe that no one can look at $u_{1t}^{\ x}$ and figure out what it is supposed to mean. So the parentheses have been added in the interest of readability.

For example, if we let t be $g(c)$ and we let u be $f(x, y) + h(z, x, g(x))$, then u_t^x is
$$f(g(c), y) + h(z, g(c), g(g(c))).$$

The definition of substitution into a formula is also by recursion:

Definition 1.8.2. Suppose that ϕ is an $\mathcal{L}$-formula, t is a term, and x is a variable. We define the formula ϕ_t^x (read "ϕ with x replaced by t") as follows:

1. If $\phi :\equiv = u_1 u_2$, then ϕ_t^x is $= (u_1)_t^x (u_2)_t^x$.
2. If $\phi :\equiv R u_1 u_2 \dots u_n$, then ϕ_t^x is $R(u_1)_t^x (u_2)_t^x \dots (u_n)_t^x$.
3. If $\phi :\equiv \neg(\alpha)$, then ϕ_t^x is $\neg(\alpha_t^x)$.

4. If $\phi :\equiv (\alpha \vee \beta)$, then ϕ_t^x is $(\alpha_t^x \vee \beta_t^x)$.

5. If $\phi :\equiv (\forall y)(\alpha)$, then

$$\phi_t^x = \begin{cases} \phi & \text{if } x \text{ is } y \\ (\forall y)(\alpha_t^x) & \text{otherwise.} \end{cases}$$

As an example, suppose that ϕ is the formula

$$P(x, y) \to \big[(\forall x)(Q(g(x), z)) \vee (\forall y)(R(x, h(x))\big].$$

Then, if t is the term $g(c)$, we get

$$\phi_t^x \text{ is } P(g(c), y) \to \big[(\forall x)(Q(g(x), z)) \vee (\forall y)(R(g(c), h(g(c)))\big].$$

Having defined what we mean when we substitute a term for a variable, we will now define what it means for a term to be substitutable for a variable in a formula. The idea is that if t is substitutable for x in ϕ, we will not run into the problems discussed at the beginning of this section—we will not substitute a term in such a way that a variable contained in that term is inadvertently bound by a quantifier.

Definition 1.8.3. Suppose that ϕ is an $\mathcal{L}$-formula, t is a term, and x is a variable. We say that t **is substitutable for** x **in** ϕ if

1. ϕ is atomic, or

2. $\phi :\equiv \neg(\alpha)$ and t is substitutable for x in α, or

3. $\phi :\equiv (\alpha \vee \beta)$ and t is substitutable for x in both α and β, or

4. $\phi :\equiv (\forall y)(\alpha)$ and either

   (a) x is not free in ϕ, or

   (b) y does not occur in t and t is substitutable for x in α.

Notice that ϕ_t^x is defined whether or not t is substitutable for x in ϕ. Usually, we will not want to do a substitution unless we check for substitutability, but we have the ability to substitute whether or not it is a good idea. In the next chapter, however, you will often see that certain operations are allowed only if t is substitutable for x in ϕ. That restriction is there for good reason, as we will be concerned with preserving the truth of formulas after performing substitutions.

1.8.1 Exercises

1. For each of the following, write out u_t^x:

   (a) $u :\equiv \cos x$, t is $\sin y$.

   (b) $u :\equiv y$, t is Sy.

   (c) $u :\equiv \sharp(x, y, z)$, t is $423 - w$.

2. For each of the following, first write out ϕ_t^x, then decide if t is substitutable for x in ϕ, and then (if you haven't already) use the definition of substitutability to justify your conclusions.

   (a) $\phi :\equiv \forall x(x = y \to Sx = Sy)$, t is $S0$.

   (b) $\phi :\equiv \forall y(x = y \to Sx = Sy)$, t is Sy.

   (c) $\phi :\equiv x = y \to (\forall x)(Sx = Sy)$, t is Sy.

3. Show that if t is variable-free, then t is always substitutable for x in ϕ.

4. Show that x is always substitutable for x in ϕ.

5. Prove that if x is not free in ψ, then ψ_t^x is ψ.

6. You might think that $(\phi_y^x)_x^y$ is ϕ, but a moment's thought will give you an example to show that this doesn't always work. (What if y is free in ϕ?) Find an example that shows that even if y is not free in ϕ, we can still have $(\phi_y^x)_x^y$ different from ϕ. Under what conditions do we know that $(\phi_y^x)_x^y$ is ϕ?

7. Write a computer program (in your favorite language, or in pseudocode) that accepts as input a formula ϕ, a variable x, and a term t and outputs "yes" or "no" depending on whether or not t is substitutable for x in ϕ.

1.9 Logical Implication

At first glance it seems that a large portion of mathematics can be broken down into answering questions of the form: If I know this statement is true, is it necessarily the case that this other statement is true? In this section we will formalize that question.

Definition 1.9.1. Suppose that Δ and Γ are sets of $\mathcal{L}$-formulas. We will say that Δ **logically implies** Γ and write $\Delta \models \Gamma$ if for every $\mathcal{L}$-structure $\mathfrak{A}$, if $\mathfrak{A} \models \Delta$, then $\mathfrak{A} \models \Gamma$.

This definition is a little bit tricky. It says that if Δ is true in $\mathfrak{A}$, then Γ is true in $\mathfrak{A}$. Remember, for Δ to be true in $\mathfrak{A}$, it must be the case that $\mathfrak{A} \models \Delta[s]$ for *every* assignment function s. See Exercise 4.

If $\Gamma = \{\gamma\}$ is a set consisting of a single formula, we will write $\Delta \models \gamma$ rather than the official $\Delta \models \{\gamma\}$.

Definition 1.9.2. An $\mathcal{L}$-formula ϕ is said to be **valid** if $\emptyset \models \phi$, in other words, if ϕ is true in every $\mathcal{L}$-structure with every assignment function s. In this case, we will write $\models \phi$.

Chaff: It doesn't seem like it would be easy to check whether $\Delta \models \Gamma$. To do so directly would mean that we would have to examine every possible $\mathcal{L}$-structure and every possible assignment function s, of which there will be many.

I'm also sure that you've noticed that this double turnstyle symbol, $\models$, is getting a lot of use. Just remember that if there is a structure on the left, $\mathfrak{A} \models \sigma$, we are discussing truth in a single structure. If there is a set of sentences on the left, $\Gamma \models \sigma$, then we are discussing logical implication.

Example 1.9.3. Let $\mathcal{L}$ be the language consisting of a single binary relation symbol, P, and let σ be the sentence $(\exists y \forall x P(x, y)) \rightarrow (\forall x \exists y P(x, y))$. We show that σ is valid.

So let $\mathfrak{A}$ be any $\mathcal{L}$-structure and let $s : \mathit{Vars} \rightarrow A$ be any assignment function. We must show that

$$\mathfrak{A} \models \Big[(\exists y \forall x P(x, y)) \rightarrow (\forall x \exists y P(x, y))\Big][s].$$

Assume that $\mathfrak{A} \models (\exists y \forall x P(x, y))\,[s]$. (If $\mathfrak{A}$ does not model this sentence, then we know by the definition of $\rightarrow$ that $\mathfrak{A} \models \sigma[s]$.)

Since we know that $\mathfrak{A} \models (\exists y \forall x P(x, y))\,[s]$, we know that there is an element of the universe, a, such that $\mathfrak{A} \models \forall x P(x, y)[s[y|a]]$. And so, again by the definition of satisfaction, we know that if b is any element of A, $\mathfrak{A} \models P(x, y)\,[(s[y|a])\,[x|b]]$. If we chase through the definition of satisfaction (Definition 1.7.4) and of the various assignment functions, this means that for our one fixed a, the ordered pair $(b, a) \in P^{\mathfrak{A}}$ for any choice of $b \in A$, .

We have to prove that $\mathfrak{A} \models (\forall x \exists y P(x, y))\,[s]$. As the statement of interest is universal, we must show that, if c is an arbitrary element of A, $\mathfrak{A} \models \exists y P(x, y)[s[x|c]]$, which means that we must produce an element of the universe, d, such that $\mathfrak{A} \models P(x, y)\,[(s[x|c])\,[y|d]]$. Again, from the definition of satisfaction this means that we must find a $d \in A$ such that $(c, d) \in P^{\mathfrak{A}}$. Fortunately, we have such a d in hand, namely a. As we know $(c, a) \in P^{\mathfrak{A}}$, we have shown $\mathfrak{A} \models (\forall x \exists y P(x, y))\,[s]$, and we are finished.

1.9.1 Exercises

1. Show that $\{\alpha, \alpha \to \beta\} \models \beta$ for any formulas α and β. Translate this result into everyday English. Or Norwegian, if you prefer.

2. Show that the formula $x = x$ is valid. Show that the formula $x = y$ is not valid. What can you prove about the formula $\neg x = y$ in terms of validity?

3. Suppose that ϕ is an $\mathcal{L}$-formula and x is a variable. Prove that ϕ is valid if and only if $(\forall x)(\phi)$ is valid. Thus, if ϕ has free variables x, y, and z, ϕ will be valid if and only if $\forall x \forall y \forall z \phi$ is valid. The sentence $\forall x \forall y \forall z \phi$ is called the **universal closure** of ϕ.

4. (a) Assume that $\models (\phi \to \psi)$. Show that $\phi \models \psi$.

   (b) Suppose that ϕ is $x < y$ and ψ is $z < w$. Show that $\phi \models \psi$ but $\not\models (\phi \to \psi)$. (The slash through $\models$ means "does not logically imply.")

   [This exercise shows that the two possible ways to define logical equivalence are not equivalent. The strong form of the definitions says that ϕ and ψ are logically equivalent if $\models (\phi \to \psi)$ and $\models (\psi \to \phi)$. The weak form of the definition states that ϕ and ψ are logically equivalent if $\phi \models \psi$ and $\psi \models \phi$.]

1.10 Summing Up, Looking Ahead

What we have tried to do in this first chapter is to introduce the concepts of formal languages and formal structures. We hope that you will agree that you have seen many mathematical structures in the past, even though you may not have called them structures at the time. By formalizing what we mean when we say that a formula is true in a structure, we will be able to tie together truth and provability in the next couple of chapters.

You might be at a point where you are about to throw your hands up in disgust and say, "Why does any of this matter? I've been doing mathematics for over ten years without worrying about structures or assignment functions, and I have been able to solve problems and succeed as a mathematician so far." Allow us to assure you that the effort and the almost unreasonable precision that we are imposing on our exposition will have a payoff in later chapters. The major theorems that we wish to prove are theorems about the existence or nonexistence of certain objects. To prove that you cannot express a certain idea in a certain language, we have to *know*, with an amazing amount of exactitude, what a language is and what structures are. Our goals are some theorems that are easy to state incorrectly, so by being precise about what we are saying, we will be able to make (and prove) claims that are truly revolutionary.

Since we will be talking about the existence and nonexistence of proofs, we now must turn our attention to defining (yes, precisely) what sorts of things qualify as proofs. That is the topic of the next chapter.

Chapter 2

Deductions

2.1 Naïvely

What is it that makes mathematics different from other academic subjects? What is it that distinguishes a mathematician from a poet, a linguist, a biologist, or a civil engineer? We are sure that you have many answers to that question, not all of which are complimentary to the authors of this work or to the mathematics instructors that you have known!

We would like to suggest that one of the things that sets mathematics apart is the insistence upon proof. Mathematical statements are not accepted as true until they have been verified, and verified in a very particular manner. This process of verification is central to the subject and serves to define our field of study in the minds of many. Allow us to quote a famous story from John Aubrey's *Brief Lives*:

> [Thomas Hobbes] was 40 years old before he looked on Geometry; which happened accidentally. Being in a Gentleman's Library, Euclid's Elements lay open and 'twas the 47 El. libri 1 [the Pythagorean Theorem]. He read the Proposition. By G—, sayd he (he would now and then sweare an emphaticall Oath by way of emphasis) this is impossible! So he reads the Demonstration of it, which referred him back to such a Proposition; which proposition he read. That referred him back to another, which he also read. Et sic deinceps [and so on] that at last he was demonstratively convinced of that trueth. This made him in love with Geometry.

Doesn't this match pretty well with your image of a mathematical proof? To prove a proposition, you start from some first principles, derive some results from those axioms, then, using those axioms and results, push on to prove other results. This is a technique that you have seen in geometry courses, college mathematics courses, and in the first chapter of this book.

Our goal in this chapter will be to define, precisely, something called a deduction. You probably haven't seen a deduction before, and you aren't going to see very many of them after this chapter is over, but our idea will be that any mathematical proof should be able to be translated into a (probably very long) deduction. This will be crucial in our interpretation of the results of Chapters 3 and 5, where we will discuss the existence and nonexistence of certain deductions, and interpret those results as making claims about the existence and nonexistence of mathematical proofs.

If you think about what a proof is, you probably will come up with a characterization along the lines of: A proof is a sequence of statements, each one of which can be justified by referring to previous statements. This is a perfectly reasonable starting point, and it brings us to the main difficulty we will have to address as we move from an informal understanding of what constitutes a proof to a formal definition of a deduction: What do you mean by the word *justified*?

Our answer to this question will come in three parts. We will start by specifying a set Λ of $\mathcal{L}$-formulas, which will be called the logical axioms. Logical axioms will be deemed to be "justified" in any deduction. Depending on the situation at hand, we will then specify a set of nonlogical axioms, Σ. Finally, we will develop some rules of inference, which will be ordered pairs (Γ, ϕ), where Γ is a finite set of formulas and ϕ is a formula. Then, if α is a formula, we will say that a deduction of α from Σ is a finite list of formulas $\phi_1, \phi_2, \ldots, \phi_n$ such that ϕ_n is α and for each i, ϕ_i is justified by virtue of being either a logical axiom ($\phi_i \in \Lambda$), a nonlogical axiom ($\phi_i \in \Sigma$), or the conclusion of one of our rules of inference, (Γ, ϕ_i), where $\Gamma \subseteq \{\phi_1, \phi_2, \ldots, \phi_{i-1}\}$.

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

In order to do this, we will impose the following restrictions on our logical axioms and rules of inference:

1. There will be an algorithm (i.e., a mechanical procedure) that will decide, given a formula θ, whether or not θ is a logical axiom.

2. There will be an algorithm that will decide, given a finite set of formulas Γ and a formula θ, whether or not (Γ, θ) is a rule of inference.

3. For each rule of inference (Γ, θ), Γ will be a finite set of formulas.

4. Each logical axiom will be valid.

5. Our rules of inference will preserve truth. In other words, for each rule of inference (Γ, θ), $\Gamma \models \theta$.

The idea here is that although it may require no end of brilliance and insight to discover a deduction of a formula α, there should be *no* brilliance and *no* insight required to check whether an alleged deduction of α is, in fact, a deduction of α. To check whether a deduction is correct will be such a simple procedure that it could be programmed into a computer. Furthermore, we will be certain that if a deduction of α from Σ is given, and if we look at a mathematical structure $\mathfrak{A}$ such that $\mathfrak{A} \models \Sigma$, then we will be certain that $\mathfrak{A} \models \alpha$. This is what we mean when we say that our deductions will preserve truth.

2.2 Deductions

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

Definition 2.2.1. Suppose that Σ is a collection of $\mathcal{L}$-formulas and D is a finite sequence $(\phi_1, \phi_2, \ldots, \phi_n)$ of $\mathcal{L}$-formulas. We will say that D is a **deduction from** Σ if for each i, $1 \leq i \leq n$, either

1. $\phi_i \in \Lambda$ (ϕ_i is a logical axiom), or
2. $\phi_i \in \Sigma$ (ϕ_i is a nonlogical axiom), or
3. There is a rule of inference (Γ, ϕ_i) such that $\Gamma \subseteq \{\phi_1, \phi_2, \ldots, \phi_{i-1}\}$.

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

> *Chaff:* Well, we have now established what we mean by the word *justified*. In a deduction we are allowed to write down any $\mathcal{L}$-formula that we like, as long as that formula is either a logical axiom or is listed explicitly in a collection Σ of nonlogical axioms. Any formula that we write in a deduction that is *not* an axiom must arise from previous formulas in the deduction via a rule of inference.
>
> You may have gathered that there are many different deductive systems, depending on the choices that are made for Λ, and the rules of inference. As a general rule, a deductive system will either have lots of rules of inference and few logical axioms, or

not too many rules and a lot of axioms. In developing the deductive system for us to use in this book, we attempt to pursue a middle course.

Also notice that $\vdash$ is another metalinguistic symbol. It is not part of the language $\mathcal{L}$.

Example 2.2.2. Suppose, for starters, that we don't want to make any assumptions. So, let $\Sigma = \emptyset$, let $\Lambda = \emptyset$, and write down a deduction from Σ. Don't be shy. Go ahead. We'll wait.

Still nothing? Right. There are no deductions from the empty set of axioms. (Actually, after we set up our rules of inference, there will be some deductions from the empty set of axioms, but that comes later.) This is a problem that the English logician Bertrand Russell found particularly annoying as he began to learn mathematics.

> At the age of eleven, I began Euclid, with my brother as my tutor. This was one of the great events of my life, as dazzling as first love. I had not imagined that there was anything so delicious in the world. ... From that moment until Whitehead and I finished *Principia Mathematica*, when I was thirty-eight, mathematics was my chief interest, and my chief source of happiness. Like all happiness, however, it was not unalloyed. I had been told that Euclid proved things, and was much disappointed that he started with axioms. At first I refused to accept them unless my brother could offer me some reason for doing so, but he said: "If you don't accept them we cannot go on," and as I wished to go on, I reluctantly admitted them *pro tem*. The doubt as to the premisses of mathematics which I felt at that moment remained with me, and determined the course of my subsequent work. [Russell 67, p. 36]

What we have managed to do with our definition of deduction, though, is to be up front about our need to make assumptions, and we will acknowledge our axiom set in every deduction that we write.

Example 2.2.3. Let us work in the language $\mathcal{L} = \{P\}$, where P is a binary

relation symbol. Let Σ, our set of axioms, be

$$\begin{aligned}\Sigma = \{&\forall x P(x,x),\\ &P(u,v),\\ &P(u,v) \to P(v,u),\\ &P(v,u) \to P(u,u)\}.\end{aligned}$$

We will let $\Lambda = \emptyset$ for now. We also need to have a set of rules of inference. So temporarily let our set of rules of inference be

$$\big\{\big(\{\alpha, \alpha \to \beta\}, \beta\big) \mid \alpha \text{ and } \beta \text{ are formulas of } \mathcal{L}\big\}.$$

This is just the rule modus ponens, which says that from the formulas α and $\alpha \to \beta$ we may conclude β.

Now we can write a deduction from Σ of the formula $P(u,u)$, as follows:

$$\begin{aligned}&P(u,v)\\ &P(u,v) \to P(v,u)\\ &P(v,u)\\ &P(v,u) \to P(u,u)\\ &P(u,u).\end{aligned}$$

You can easily see that every formula in our deduction is either explicitly listed among the elements of our axiom set Σ, or follows from modus ponens from previously listed formulas in the deduction.

Notice, however, that we cannot use the universal statement $\forall x P(x,x)$ to derive our needed formula $P(u,u)$. Even a statement that seems like it ought to follow from our axioms, $P(v,v)$, for example, will not be deducible from Σ until we either add to our rules of inference or include some additional axioms. Our definition of a deduction is very limiting—we cannot even use standard logical tricks such as universal instantiation [from $\forall x\, blah(x)$ deduce $blah(t)$]. These logical axioms will be gathered together in Section 2.3.

> *Chaff:* It is really tempting here to write down the incorrect deduction
>
> $$\begin{aligned}&\forall x P(x,x)\\ &P(u,u).\end{aligned}$$
>
> Please don't say things like that until we have built our collection of logical axioms. Remember, what we are trying to do here is to have a definition of deduction that is entirely syntactic, that does not depend on the meanings of the symbols. Where you are likely to run into trouble is when you start thinking too much about the meanings of the things that you write

down. Our definition gives us deductions that are easily verifiable: Given an alleged deduction from Σ, as long as we can decide what formulas are in Σ, we can decide if the alleged deduction is correct. In fact, we could easily program a computer to check the deduction for us. However, this ease in verification comes with a price: Deductions are difficult to write and hard to motivate.

Definition 2.2.1 is a "bottom-up" definition. It defines a deduction in terms of its parts. Another way to define a collection of things is to take a "top-down" approach. The next proposition does just that, by showing that we can think of the collection of deductions from Σ (called Thm_Σ) as the closure of the collection of axioms under the application of the rules of inference.

Proposition 2.2.4. *Fix sets of $\mathcal{L}$-formulas Σ and Λ and a collection of rules of inference. The set $Thm_\Sigma = \{\phi \mid \Sigma \vdash \phi\}$ is the smallest set C such that*

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

Proof. This proposition makes two separate claims about the set Thm_Σ. The first claim is that Thm_Σ satisfies the three criteria. The second claim is that Thm_Σ is the *smallest* set to satisfy the criteria. We tackle these claims one at a time.

First, let us look at the criteria in order, and make sure that Thm_Σ satisfies them. So to begin, we must show that $\Sigma \subseteq \mathrm{Thm}_\Sigma$. But certainly if $\sigma \in \Sigma$, there is a deduction-from-Σ of σ, for example this one-line deduction: σ. Similarly, to show that $\Lambda \subseteq \mathrm{Thm}_\Sigma$, we notice that there is a one-line deduction of any $\lambda \in \Lambda$. To finish this part of the proof, we must show that if (Γ, θ) is a rule of inference and $\Gamma \subseteq \mathrm{Thm}_\Sigma$, then $\theta \in \mathrm{Thm}_\Sigma$. But to produce a deduction-from-Σ of θ, all we have to do is write down deductions of each of the γ's in Γ, followed by the formula θ. This is a valid deduction, as θ follows from Γ by the rule of inference (Γ, θ). Thus Thm_Σ satisfies the three criteria of the proposition.

Now we must show that Thm_Σ is the smallest such set. This is quite easy to prove once you figure out what you have to do. What is claimed is that if C is a collection of formulas satisfying the given requirements, then $\mathrm{Thm}_\Sigma \subseteq C$. So we assume that C is a class satisfying the conditions, and we attempt to show that every element of Thm_Σ is in C.

If $\phi \in \mathrm{Thm}_\Sigma$, there is a deduction from Σ with last line ϕ. If the entry ϕ is justified by virtue of ϕ being either a logical or nonlogical axiom, then ϕ is explicitly included in the set C. If ϕ is justified by reference to a rule

of inference (Γ, ϕ), then each $\gamma \in \Gamma$ is an element of C (this is really a proof by induction, and here is where we use the inductive hypothesis), and thus, by the third requirement on C, $\phi \in C$, as needed.

Since $\mathrm{Thm}_\Sigma \subseteq C$ for all such sets C, Thm_Σ is the *smallest* such set, as claimed. □

Here is what we will do in the next few sections: We will define Λ, the fixed set of logical axioms; we will establish our collection of rules of inference; we will prove some results about deductions; and finally, we will discuss some examples of sets of nonlogical axioms.

2.2.1 Exercises

1. Let the collection of nonlogical axioms be

$$\Sigma = \{(A(x) \wedge A(x)) \to B(x, y), A(x), B(x, y) \to A(x)\},$$

and let the rule of inference be modus ponens, as in Example 2.2.3. For each of the following, decide if it is a deduction. If it is not a deduction, explain how you know that it is not a deduction.

(a)
$$\begin{array}{l} A(x) \\ A(x) \wedge A(x) \\ (A(x) \wedge A(x)) \to B(x, y) \\ B(x, y) \end{array}$$

(b)
$$\begin{array}{l} B(x, y) \to A(x) \\ A(x) \\ B(x, y) \end{array}$$

(c)
$$\begin{array}{l} (A(x) \wedge A(x)) \to B(x, y) \\ B(x, y) \to A(x) \\ (A(x) \wedge A(x)) \to A(x) \end{array}$$

2. Consider the axiom system Σ of Example 2.2.3. It is implied in that example that there is no deduction from Σ of the formula $P(v, v)$. Prove this fact.

3. Carefully write out the proof of Proposition 2.2.4, worrying about the inductive step. [*Suggestion:* You may want to proceed by induction on the length of the shortest deduction of ϕ.]

4. Let $\mathcal{L}$ be a language that consists of a single unary predicate symbol R, and let B be the infinite set of axioms

$$\begin{aligned} B = \{&R(x_1),\\ &R(x_1) \to R(x_2),\\ &R(x_2) \to R(x_3),\\ &\qquad\vdots\\ &R(x_i) \to R(x_{i+1}),\\ &\qquad\vdots\\ &\qquad\}. \end{aligned}$$

Using modus ponens as the only rule of inference, prove by induction that $B \vdash R(x_j)$ for each natural number $j \geq 1$.

2.3 The Logical Axioms

Let a first-order language $\mathcal{L}$ be given. In this section we will gather together a collection Λ of logical axioms for $\mathcal{L}$. This set of axioms, though infinite, will be decidable. Roughly this means that if we are given a formula ϕ that is alleged to be an element of Λ, we will be able to decide whether $\phi \in \Lambda$ or $\phi \notin \Lambda$. Furthermore, we could, in principle, design a computer program that would be able to decide membership in Λ in a finite amount of time.

After we have established the set of logical axioms Λ and we want to start doing mathematics, we will want to add additional axioms that are designed to allow us to deduce statements about whatever mathematical system we may have in mind. These will constitute the collection of nonlogical axioms, Σ. For example, if we are working in number theory, using the language $\mathcal{L}_{NT}$, along with the logical axioms Λ we will also want to use other axioms that concern the properties of addition and the ordering relation denoted by the symbol $<$. These additional axioms are the formulas that we will place in Σ. Then, from this expanded set of axioms $\Lambda \cup \Sigma$ we will attempt to write deductions of formulas that make statements of number-theoretic interest. To reiterate: Λ, the set of logical axioms, will be fixed, as will the collection of rules of inference. But the set of nonlogical axioms must be specified for each deduction. In the current section we set out the logical axioms only, dealing with the rules of inference in Section 2.4, and deferring our discussion of the nonlogical axioms until Section 2.8.

2.3.1 Equality Axioms

We have taken the route of assuming that the equality symbol, $=$, is a part of the language $\mathcal{L}$. There are three groups of axioms that are designed for this symbol. The first just says that any object is equal to itself:

$$x = x \text{ for each variable } x. \tag{E1}$$

For the second group of axioms, assume that $x_1, x_2, \ldots, x_n$ are variables, $y_1, y_2, \ldots, y_n$ are variables, and f is an n-ary function symbol.

$$\big[(x_1 = y_1) \wedge (x_2 = y_2) \wedge \cdots \wedge (x_n = y_n)\big] \rightarrow (f(x_1, x_2, \ldots, x_n) = f(y_1, y_2, \ldots, y_n)). \tag{E2}$$

The assumptions for the third group of axioms is the same as for the second group, except that R is assumed to be an n-ary relation symbol (R might be the equality symbol, which is seen as a binary relation symbol).

$$\big[(x_1 = y_1) \wedge (x_2 = y_2) \wedge \cdots \wedge (x_n = y_n)\big] \rightarrow (R(x_1, x_2, \ldots, x_n) \rightarrow R(y_1, y_2, \ldots, y_n)). \tag{E3}$$

Axioms (E2) and (E3) are axioms that are designed to allow substitution of equals for equals. Nothing fancier than that.

2.3.2 Quantifier Axioms

The quantifier axioms are designed to allow a very reasonable sort of entry in a deduction. Suppose that we know $\forall x P(x)$. Then, if t is any term of the language, we should be able to state $P(t)$. To avoid problems of the sort outlined at the beginning of Section 1.8, we will demand that the term t be substitutable for the variable x.

$$(\forall x \phi) \rightarrow \phi_t^x, \text{ if } t \text{ is substitutable for } x \text{ in } \phi. \tag{Q1}$$

$$\phi_t^x \rightarrow (\exists x \phi), \text{ if } t \text{ is substitutable for } x \text{ in } \phi. \tag{Q2}$$

In many logic texts, axiom (Q1) would be called universal instantiation, while (Q2) would be known as existential generalization. We will avoid this impressive language and stick with the more mundane (Q1) and (Q2).

2.3.3 Recap

Just to gather all of the logical axioms together in one place, let us state them once again. The set Λ of logical axioms is the collection of all formulas that fall into one of the following categories:

$$x = x \text{ for each variable } x. \tag{E1}$$

$$\big[(x_1 = y_1) \wedge (x_2 = y_2) \wedge \cdots \wedge (x_n = y_n)\big] \rightarrow (f(x_1, x_2, \ldots, x_n) = f(y_1, y_2, \ldots, y_n)). \tag{E2}$$

$$\big[(x_1 = y_1) \wedge (x_2 = y_2) \wedge \cdots \wedge (x_n = y_n)\big] \rightarrow (R(x_1, x_2, \ldots, x_n) \rightarrow R(y_1, y_2, \ldots, y_n)). \tag{E3}$$

$$(\forall x \phi) \rightarrow \phi_t^x, \text{ if } t \text{ is substitutable for } x \text{ in } \phi. \tag{Q1}$$

$$\phi_t^x \rightarrow (\exists x \phi), \text{ if } t \text{ is substitutable for } x \text{ in } \phi. \tag{Q2}$$

Notice that Λ is decidable: We could write a computer program which, given a formula ϕ, can decide in a finite amount of time whether or not ϕ is an element of Λ.

2.4 Rules of Inference

Having established our set Λ of logical axioms, we must now fix our rules of inference. There will be two types of rules, one dealing with propositional consequence and one dealing with quantifiers.

2.4.1 Propositional Consequence

In all likelihood you are familiar with tautologies of propositional logic. They are simply formulas like $(A \rightarrow B) \leftrightarrow (\neg B \rightarrow \neg A)$. If you are comfortable with tautologies, feel free to skip over the next couple of paragraphs. If not, what follows is a very brief review of a portion of propositional logic.

We work with a restricted language $\mathcal{P}$, consisting only of a set of propositional variables $A, B, C, \ldots$ and the connectives $\vee$ and $\neg$. Notice there are no quantifiers, no relation symbols, no function symbols, and no constants. Formulas of propositional logic are defined as being the collection of all ϕ such that either ϕ is a propositional variable, or ϕ is $(\neg\alpha)$, or ϕ is $(\alpha \vee \beta)$, with α and β being formulas of propositional logic.

Each propositional variable can be assigned one of two truth values, T or F, corresponding to truth and falsity. Given such an assignment (which is really a function v : propositional variables $\rightarrow \{T, F\}$), we can extend v to a function $\overline{v}$ assigning a truth value to any propositional formula as follows:

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

Now we say that a propositional formula ϕ is a tautology if and only if $\overline{v}(\phi) = T$ for any truth assignment v.

One way that you can check whether a given ϕ is a tautology is by constructing a truth table with one row for each possible combination of truth values for the propositional variables that occur in ϕ. Then you fill in the truth table and see whether the truth value associated with the main connective is always true. For example, consider the propositional formula $A \to (B \to A)$, which is translated to $\neg A \vee (\neg B \vee A)$. The truth table verifying that this formula is a tautology is

A	B	$\neg A$	$\vee$	$(\neg B$	$\vee$	$A)$
T	T	F	T	F	T	T
T	F	F	T	T	T	T
F	T	T	T	F	F	F
F	F	T	T	T	T	F

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

1. Find all subformulas of β of the form $\forall x \alpha$ that are not in the scope of another quantifier. Replace them with propositional variables in a systematic fashion. This means that if $\forall y Q(y, c)$ appears twice in β, it is replaced by the same letter both times, and distinct subformulas are replaced with distinct letters.

2. Find all atomic formulas that remain, and replace them systematically with new propositional variables.

3. At this point, β will have been replaced with a propositional formula β_P.

For example, suppose that we look at the $\mathcal{L}$-formula

$$(\forall x P(x) \wedge Q(c, z)) \to (Q(c, z) \vee \forall x P(x)) .$$

For the first step of the procedure above, we replace the quantified subformulas with the propositional letter B:

$$(B \wedge Q(c, z)) \to (Q(c, z) \vee B) .$$

To finish the transformation to a propositional formula, replace the atomic formula with a propositional letter:

$$(B \wedge A) \to (A \vee B) .$$

Notice that if β_P is a tautology, then β is valid, but the converse of this statement fails. For example, if β is

$$\big[(\forall x)(\theta) \wedge (\forall x)(\theta \to \rho)\big] \to (\forall x)(\rho),$$

then β is valid, but β_P would be $[A \wedge B] \to C$, which is certainly not a tautology.

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

Definition 2.4.1. Suppose that Γ_P is a set of propositional formulas and ϕ_P is a propositional formula. We will say that ϕ_P is a **propositional consequence of** Γ_P if every truth assignment that makes each propositional formula in Γ_P true also makes ϕ_P true. Notice that ϕ_P is a tautology if and only if ϕ_P is a propositional consequence of $\emptyset$.

Lemma 2.4.2. *If $\Gamma_P = \{\gamma_{1P}, \gamma_{2P}, \ldots, \gamma_{nP}\}$ is a nonempty finite set of propositional formulas and ϕ_P is a propositional formula, then ϕ_P is a propositional consequence of Γ_P if and only if*

$$[\gamma_{1P} \wedge \gamma_{2P} \wedge \cdots \wedge \gamma_{nP}] \to \phi_P$$

is a tautology.

Proof. Exercise 3. □

Now we extend our definition of propositional consequence to include formulas of first-order logic:

Definition 2.4.3. Suppose that Γ is a finite set of $\mathcal{L}$-formulas and ϕ is an $\mathcal{L}$-formula. We will say that ϕ is a **propositional consequence of** Γ if ϕ_P is a propositional consequence of Γ_P, where ϕ_P and Γ_P are the results of applying the procedure on the preceding page uniformly to ϕ and all of the formulas in Γ.

Example 2.4.4. Suppose that $\mathcal{L}$ contains two unary relation symbols, P and Q. Let Γ be the set

$$\{\forall x P(x) \to \exists y Q(y), \exists y Q(y) \to P(x), \neg P(x) \leftrightarrow (y = z)\}.$$

If we let ϕ be the formula $\forall x P(x) \to \neg(y = z)$, then by applying our procedure uniformly to the elements of Γ and ϕ, we see that

$$\Gamma_P \text{ is } \{A \to B, B \to C, \neg C \leftrightarrow D\}$$

and ϕ_P is $A \to \neg D$, where the fact that we have substituted the same propositional variables for the same formulas in ϕ and the elements of Γ is ensured by our applying the procedure *uniformly* to all of the formulas in question. At this point it is easy to verify that ϕ is a propositional consequence of Γ.

Finally, our rule of inference:

Definition 2.4.5. If Γ is a finite set of $\mathcal{L}$-formulas, ϕ is an $\mathcal{L}$-formula, and ϕ is a propositional consequence of Γ, then (Γ, ϕ) is a **rule of inference of type (PC)**.

> *Chaff:* All of this formalism just might be hiding what is really going on here. What rule (PC) says is that if you have proved γ_1 and γ_2 and $[(\gamma_1 \wedge \gamma_2) \to \phi]_P$ is a tautology, then you may conclude ϕ. Nothing fancier than that.
>
> Also notice that if ϕ is a formula such that ϕ_P is a tautology, rule (PC) allows us to assert ϕ in any deduction, using $\Gamma = \emptyset$.

2.4.2 Quantifier Rules

The motivation behind our quantifier rules is very simple. Suppose, without making any particular assumptions about x, that you were able to prove "x is an ambitious aardvark." Then it seems reasonable to claim that you have proved "$(\forall x)x$ is an ambitious aardvark." Dually, if you were able to prove the Riemann Hypothesis from the assumption that "x is a bossy bullfrog," then from the assumption "$(\exists x)x$ is a bossy bullfrog," you should still be able to prove the Riemann Hypothesis.

Definition 2.4.6. Suppose that the variable x is not free in the formula ψ. Then both of the following are **rules of inference of type (QR)**:

$$\big(\{\psi \to \phi\}, \psi \to (\forall x \phi)\big)$$
$$\big(\{\phi \to \psi\}, (\exists x \phi) \to \psi\big).$$

The "not making any particular assumptions about x" comment is made formal by the requirement that x not be free in ψ.

> *Chaff:* Just to make sure that you are not lost in the brackets of the definition, what we are saying here is that if x is not free in ψ:
>
> 1. From the formula $\psi \to \phi$, you may deduce $\psi \to (\forall x \phi)$.
> 2. From the formula $\phi \to \psi$, you may deduce $(\exists x \phi) \to \psi$.

2.4.3 Exercises

1. We claim that the collection Λ of logical axioms is decidable. Outline an algorithm which, given an $\mathcal{L}$-formula θ, outputs "yes" if θ is an element of Λ and outputs "no" if θ is not an element of Λ. You do not have to be too fussy. Notice that you have to be able to decide if a term t is substitutable for a variable x is a formula ϕ. See Exercise 7 in Section 1.8.1.

2. Show that the set of rules of inference is decidable. So outline an algorithm that will decide, given a finite set of formulas Γ and a formula θ, whether or not (Γ, θ) is a rule of inference.

3. Prove Lemma 2.4.2.

4. Write a deduction of the second quantifier axiom (Q2) (on page 49) without using (Q2) as an axiom.

5. For each of the following, decide if ϕ is a propositional consequence of Γ and justify your assertion.

   (a) Γ is $\{(\forall x P(x)) \to Q(y), (\forall x P(x)) \vee (\forall x R(x)), \exists x \neg R(x)\}$; ϕ is $Q(y)$.

   (b) Γ is $\{x = y \wedge Q(y), Q(y) \vee x + y < z\}$; ϕ is $x + y < z$.

   (c) Γ is $\{P(x, y, x), x < y \vee M(w, p), (\neg P(x, y, x)) \wedge (\neg x < y)\}$; ϕ is $\neg M(w, p)$.

6. Prove that if θ is not valid, then θ_P is not a tautology. Deduce that if θ_P is a tautology, then θ is valid.

2.5 Soundness

Mathematicians are by nature a conservative bunch. We speak not of political or social leanings, but of their professional outlook. In particular, a mathematician likes to know that when something has been proved, it is true. In this section we will prove a theorem that shows that the logical system that we have developed has this highly desirable property. This result is called the Soundness Theorem.

Let us restate the list of requirements that we set out on page 42 for our axioms and rules of inference:

1. There will be an algorithm that will decide, given a formula θ, whether or not θ is a logical axiom.

2. There will be an algorithm that will decide, given a finite set of formulas Γ and a formula θ, whether or not (Γ, θ) is a rule of inference.

3. For each rule of inference (Γ, θ), Γ will be a finite set of formulas.

4. Each logical axiom will be valid.

5. Our rules of inference will preserve truth. In other words, for each rule of inference (Γ, θ), $\Gamma \models \theta$.

These requirements serve two purposes: They allow us to verify mechanically that an alleged deduction is in fact a deduction, and they provide the basis of the Soundness Theorem. Of course, we must first verify that the system of axioms and rules that we have set out in the preceding two sections satisfies these requirements.

That the first three requirements above are satisfied by our deduction system was noted as the axioms and rules were presented. These are the rules that are needed for deduction verification. We will discuss the last two requirements in more detail and then use those requirements to prove the Soundness Theorem.

Theorem 2.5.1. *The logical axioms are valid.*

Proof. We must check both the equality axioms and the quantifier axioms. First, consider equality axioms of type (E2). [(E1) and (E3) will be proved in the Exercises.]

> *Chaff:* Let us mention that we will use Theorem 2.6.2 in this proof. Although the presentation of that result has been delayed in order to aid the flow of the exposition, you may want to look at the statement of that theorem now so you won't be surprised when it appears.

So fix a structure $\mathfrak{A}$ and an assignment function $s : \mathit{Vars} \to A$. We must show that

$$\mathfrak{A} \models \Big(\big[(x_1 = y_1) \wedge (x_2 = y_2) \wedge \cdots \wedge (x_n = y_n) \big] \to \\ (f(x_1, x_2, \ldots, x_n) = f(y_1, y_2, \ldots, y_n)) \Big)[s].$$

As the formula in question is an implication, we may assume that the antecedent is satisfied by the pair $(\mathfrak{A}, s)$, and thus $s(x_1) = s(y_1), s(x_2) = s(y_2), \ldots,$ and $s(x_n) = s(y_n)$. We must prove that $\mathfrak{A} \models (f(x_1, x_2, \ldots, x_n) = f(y_1, y_2, \ldots, y_n))[s]$. From the definition of satisfaction (Definition 1.7.4), we know this means that we have to show

$$\overline{s}(f(x_1, x_2, \ldots, x_n)) = \overline{s}(f(y_1, y_2, \ldots, y_n)).$$

Now we look at the definition of term assignment function (Definition 1.7.3) and see that we must prove

$$f^{\mathfrak{A}}(\overline{s}(x_1), \overline{s}(x_2), \ldots, \overline{s}(x_n)) = f^{\mathfrak{A}}(\overline{s}(y_1), \overline{s}(y_2), \ldots, \overline{s}(y_n)).$$

But since $\overline{s}(x_i) = s(x_i) = s(y_i) = \overline{s}(y_i)$, and since $f^{\mathfrak{A}}$ is a function, this is true. Thus our equality axiom (E2) is valid.

Now we examine the quantifier axiom of type (Q1), reserving (Q2) for the Exercises. Once again, fix $\mathfrak{A}$ and s, and assume that the term t is substitutable for the variable x in the formula ϕ. We must show that

$$\mathfrak{A} \models \left[(\forall x\phi) \to \phi_t^x\right][s].$$

So once again, we assume that $\mathfrak{A} \models (\forall x\phi)[s]$, and we show that $\mathfrak{A} \models \phi_t^x[s]$. By assumption, $\mathfrak{A} \models \phi[s[x|a]]$ for any element $a \in A$, so in particular, $\mathfrak{A} \models \phi[s[x|\overline{s}(t)]]$.

Informally, this says that ϕ is true in $\mathfrak{A}$ with assignment function s, where you interpret x as $\overline{s}(t)$. It is plausible, given our assumption that t is substitutable for x in ϕ, that if we altered the formula ϕ by replacing x by t, then ϕ_t^x would be true in $\mathfrak{A}$ with assignment function s. This is the content of Theorem 2.6.2. Since we know that $\mathfrak{A} \models \phi[s[x|\overline{s}(t)]]$ and Theorem 2.6.2 states that this is equivalent to $\mathfrak{A} \models \phi_t^x[s]$, we have established $\mathfrak{A} \models \phi_t^x[s]$, so we have proved that axioms of type (Q1) are valid.

Thus, modulo your proofs of (E1), (E2), and (Q2) and the delayed proof of Theorem 2.6.2, all of our logical axioms are valid, and our proof is complete. □

This leaves one more item on our list of requirements to check. We must show that our rules of inference preserve truth.

Theorem 2.5.2. *Suppose that* (Γ, θ) *is a rule of inference. Then* $\Gamma \models \theta$.

Proof. First, assume that (Γ, θ) is a rule of type (PC). Then Γ is finite, and by Lemma 2.4.2, we know that

$$[\gamma_{1P} \wedge \gamma_{2P} \wedge \cdots \wedge \gamma_{nP}] \to \theta_P$$

is a tautology, where $\Gamma_P = \{\gamma_{1P}, \gamma_{2P}, \ldots, \gamma_{nP}\}$ is the set of propositional formulas corresponding to Γ and θ_P is the propositional formula corresponding to θ. But then, by Exercise 6 on page 54, we know that

$$[\gamma_1 \wedge \gamma_2 \wedge \cdots \wedge \gamma_n] \to \theta$$

is valid, and thus $\Gamma \models \theta$.

The other possibility is that our rule of inference is a quantifier rule. So, suppose that x is not free in ψ. We show that $(\psi \to \phi) \models [\psi \to (\forall x\phi)]$, leaving the other (QR) rule for the Exercises.

So fix a structure $\mathfrak{A}$ and assume that $\mathfrak{A} \models (\psi \to \phi)$. Thus our assumption is that for any assignment s, $\mathfrak{A} \models (\psi \to \phi)[s]$. We must show that $\mathfrak{A} \models (\psi \to \forall x\phi)$, which means that we must show that $(\psi \to \forall x\phi)$ is satisfied in $\mathfrak{A}$ under every assignment function. So let an assignment function $t : \mathit{Vars} \to A$ be given. We must show that $\mathfrak{A} \models (\psi \to \forall x\phi)[t]$. If $\mathfrak{A} \not\models \psi[t]$,

we are done, so assume that $\mathfrak{A} \models \psi[t]$. We want to prove that $\mathfrak{A} \models \forall x \phi[t]$, which means that if a is any element of A, we must show that $\mathfrak{A} \models \phi[t[x|a]]$.

We know, by assumption, that $\mathfrak{A} \models (\psi \to \phi)[t[x|a]]$. Furthermore, Proposition 1.7.7 tells us that $\mathfrak{A} \models \psi[t[x|a]]$, as $\mathfrak{A} \models \psi[t]$, and t and $t[x|a]$ agree on all of the free variables of ψ (x is not free in ψ by assumption). But then, by the definition of satisfaction, $\mathfrak{A} \models \phi[t[x|a]]$, and we are finished. □

We are now at a point where we can prove the Soundness Theorem. The idea behind this theorem is very simple. Suppose that Σ is a set of $\mathcal{L}$-formulas and suppose that there is a deduction of ϕ from Σ. What the Soundness Theorem tells us is that in any structure $\mathfrak{A}$ that makes all of the formulas of Σ true, ϕ is true as well.

Theorem 2.5.3 (Soundness). *If $\Sigma \vdash \phi$, then $\Sigma \models \phi$.*

Proof. Let $\text{Thm}_\Sigma = \{\phi \mid \Sigma \vdash \phi\}$, and let $C = \{\phi \mid \Sigma \models \phi\}$. We show that $\text{Thm}_\Sigma \subseteq C$, which proves the theorem.

Notice that C has the following characteristics:

1. $\Sigma \subseteq C$. If $\sigma \in \Sigma$, then certainly $\Sigma \models \sigma$.

2. $\Lambda \subseteq C$. As the logical axioms are valid, they are true in any structure. Thus $\Sigma \models \lambda$ for any logical axiom λ, which means that if $\lambda \in \Lambda$, then $\lambda \in C$, as needed.

3. If (Γ, θ) is a rule of inference and $\Gamma \subseteq C$, then $\theta \in C$. So assume that $\Gamma \subseteq C$. To prove $\theta \in C$ we must show that $\Sigma \models \theta$. Fix a structure $\mathfrak{A}$ such that $\mathfrak{A} \models \Sigma$. We must prove that $\mathfrak{A} \models \theta$.

   If γ is any element of Γ, then since $\gamma \in C$, we know that $\Sigma \models \gamma$. Since $\mathfrak{A} \models \Sigma$ and $\Sigma \models \gamma$, we know that $\mathfrak{A} \models \gamma$. But this says that $\mathfrak{A} \models \gamma$ for each $\gamma \in \Gamma$, so $\mathfrak{A} \models \Gamma$. But Theorem 2.5.2 tells us that $\Gamma \models \theta$, since (Γ, θ) is a rule of inference. Therefore, since $\mathfrak{A} \models \Gamma$ and $\Gamma \models \theta$, $\mathfrak{A} \models \theta$, as needed.

So C is a set of the type outlined in Proposition 2.2.4, and by that proposition, $\text{Thm}_\Sigma \subseteq C$, as needed. □

Notice that the Soundness Theorem begins to tie together the notions of deducibility and logical implication. It says, "If there is a deduction from Σ of ϕ, then Σ logically implies ϕ." Thus the purely syntactic notion of deduction, a notion that relies only upon typographical considerations, is linked to the notions of truth and logical implication, ideas that are inextricably tied to mathematical structures and their properties. This linkage will be tightened in Chapter 3.

> *Chaff:* The proof of the Soundness Theorem that we have presented above has the desirable qualities of being neat and quick. It emphasizes a core fact about the consequences of Σ,

namely that Thm_Σ is the smallest set of formulas satisfying the given three conditions. Unfortunately, the proof has the less desirable attribute of being pretty abstract. Exercise 5 outlines a more direct, less abstract proof of the Soundness Theorem.

2.5.1 Exercises

1. Ingrid walks into your office one day and announces that she is puzzled. She has a set of axioms Σ in the language of number theory, and she has a formula ϕ that she has proved using the assumptions in Σ. Unfortunately, ϕ is a statement that is not true in the standard model $\mathfrak{N}$. Is this a problem? If it is a problem, what possible explanations can you think of that would explain what went wrong? If it is not a problem, *why* is it not a problem?

2. Prove that the equality axioms of type (E1) and (E3) are valid.

3. Show that the quantifier axiom of type (Q2) is valid.

4. Show that, if x is not free in ψ, $(\phi \to \psi) \models [(\exists x\phi) \to \psi]$.

5. Prove the Soundness Theorem by induction on the complexity of the proof of ϕ. For the base cases, ϕ is either a logical axiom or a member of Σ. Then assume that ϕ is proved by reference to a rule of inference. Show that in this case as well, $\Sigma \models \phi$.

2.6 Two Technical Lemmas

In this section we present two rather technical lemmas that we need to complete the proof of Theorem 2.5.1. The proofs that are involved are not pretty, and if you are the trusting sort, you may want to scan through this section rather quickly. On the other hand, if you come to grips with these results, you will gain a better appreciation for the details of substitutability and assignment functions.

To motivate the first lemma, consider this example: Suppose that we are working in the language of number theory and that the structure under consideration comprises the natural numbers. Let the term u be $x \cdot v$ and the term t be $y + z$. Then u_t^x is $(y + z) \cdot v$. Now we have to fix a couple of assignment functions. Let the assignment function s look like this:

Vars	s
x	12
y	3
z	7
v	4
$\vdots$	$\vdots$

So $s(x) = 12, s(y) = 3$, and so on.

Now, suppose that s' is an assignment function that is just like s, except that s' sends x to the value $\overline{s}(t)$, which is $\overline{s}(y + z) = 3 + 7 = 10$:

Vars	s	s'
x	12	10
y	3	3
z	7	7
v	4	4
$\vdots$	$\vdots$	$\vdots$

Now, if you compare $\overline{s}(u_t^x)$ and $\overline{s'}(u)$, you find that

$$\overline{s}(u_t^x) = \overline{s}((y + z) \cdot v) = (3 + 7) \cdot 4 = 10 \cdot 4 = 40$$

$$\overline{s'}(u) = \overline{s'}(x \cdot v) = 10 \cdot 4 = 40.$$

So, in this situation, the element of the universe that is assigned by s to u_t^x is the same as the element of the universe that is assigned by s' to u. In some sense, the lemma states that it does not matter whether you alter the term or the assignment function, the result is the same.

Here is the formal statement:

Lemma 2.6.1. *Suppose that u is a term, x is a variable, and t is a term. Suppose that s : Vars $\to A$ is a variable assignment function and that $s' = s[x|\overline{s}(t)]$. Then $\overline{s}(u_t^x) = \overline{s'}(u)$.*

Proof. The proof is by induction on the complexity of the term u. If u is the variable x, then

$$\begin{aligned} \overline{s}(u_t^x) &= \overline{s}(x_t^x) \\ &= \overline{s}(t) \\ &= \overline{s'}(x) \\ &= \overline{s'}(u). \end{aligned}$$

If u is the variable y and y is different than x, then

$$\begin{aligned} \overline{s}(u_t^x) &= \overline{s}(y_t^x) \\ &= \overline{s}(y) \\ &= s(y) \\ &= s'(y) \\ &= \overline{s'}(u). \end{aligned}$$

If u is a constant symbol c, then $\overline{s}(u_t^x) = \overline{s}(c_t^x) = \overline{s}(c) = c^{\mathfrak{A}} = \overline{s'}(u)$.

The last inductive case is if u is $f(r_1, r_2, \ldots, r_n)$, with each r_i a term. In this case,

$$\begin{aligned}
\overline{s}(u_t^x) &= \overline{s}([f(r_1, r_2, \ldots, r_n)]_t^x) \\
&= \overline{s}\left(f\big((r_1)_t^x, (r_2)_t^x, \ldots, (r_n)_t^x\big)\right) \\
&= f^{\mathfrak{A}}(\overline{s}[(r_1)_t^x], \overline{s}[(r_2)_t^x], \ldots, \overline{s}[(r_n)_t^x]) && \text{definition of } \overline{s} \\
&= f^{\mathfrak{A}}(\overline{s'}(r_1), \overline{s'}(r_2), \ldots, \overline{s'}(r_n)) && \text{inductive hypothesis} \\
&= \overline{s'}(f(r_1, r_2, \ldots, r_n)) && \text{definition of } \overline{s'} \\
&= \overline{s'}(u).
\end{aligned}$$

So for every term u, $\overline{s}(u_t^x) = \overline{s'}(u)$. □

> *Chaff:* That was hard. If you understood that proof the first time through, you have done something quite out of the ordinary. If, on the other hand, you are a mere mortal, you might want to work through the proof again, keeping an example in mind as you work. Pick a language, terms u and t in your language, and a variable x. Fix a particular assignment function s. Then just follow through the steps of the proof, keeping track of where everything goes. We would write it out for you, but you will get more out of doing it for yourself. Go to it!

Our next technical result is the lemma that we quoted explicitly in the proof of Theorem 2.5.1. This theorem states that as long as t is substitutable for x in ϕ, the two different ways of evaluating the truth of "ϕ, where you interpret x as t" coincide. The first way of evaluating the truth would be by forming the formula ϕ_t^x and seeing if $\mathfrak{A} \models \phi_t^x[s]$. The second way would be to change the assignment function s to interpret x as $\overline{s}(t)$ and checking whether the original formula ϕ is true with this new assignment function. The theorem states that the two methods are equivalent.

Theorem 2.6.2. *Suppose that ϕ is an $\mathcal{L}$-formula, x is a variable, t is a term, and t is substitutable for x in ϕ. Suppose that $s : Vars \to A$ is a variable assignment function and that $s' = s[x|\overline{s}(t)]$. Then $\mathfrak{A} \models \phi_t^x[s]$ if and only if $\mathfrak{A} \models \phi[s']$.*

Proof. We use induction on the complexity of ϕ. The first base case is where $\phi :\equiv u_1 = u_2$, where u_1 and u_2 are terms. Then the following are equivalent:

$$\begin{aligned}
&\mathfrak{A} \models \phi_t^x[s] \\
&\mathfrak{A} \models (u_1)_t^x = (u_2)_t^x[s] \\
&\overline{s}\big((u_1)_t^x\big) = \overline{s}\big((u_2)_t^x\big) && \text{definition of satisfaction} \\
&\overline{s'}(u_1) = \overline{s'}(u_2) && \text{by Lemma 2.6.1} \\
&\mathfrak{A} \models \phi[s']
\end{aligned}$$

The second base case is where $\phi :\equiv R(u_1, u_2, \ldots, u_n)$. This case is similar to the case above.

The inductive cases involving the connectives $\vee$ and $\neg$ follow immediately from the inductive hypothesis.

This leaves the last inductive case, where $\phi :\equiv \forall y\psi$. We break this case down into two subcases: In the first subcase x is y, and in the second subcase x is not y.

If $\phi :\equiv \forall y\psi$ and y is x, then ϕ_t^x is ϕ. Therefore, $\mathfrak{A} \models \phi_t^x[s]$ if and only if $\mathfrak{A} \models \phi[s]$. But as s and s' agree on all of the free variables of ϕ (x is not free), by Proposition 1.7.7, $\mathfrak{A} \models \phi[s]$ if and only if $\mathfrak{A} \models \phi[s']$, as needed for this subcase.

The second subcase, where $\phi :\equiv \forall y\psi$ and y is not x, is examined in two sub-subcases:

Sub-subcase 1: If $\phi :\equiv \forall y\psi$, y is not x, and x is not free in ψ, then we know by Exercise 5 in Section 1.8.1 that ψ_t^x is ψ, and thus ϕ_t^x is ϕ. But then

$$\begin{array}{ll} \mathfrak{A} \models \phi_t^x[s] & \text{iff} \\ \mathfrak{A} \models \phi[s] & \text{iff} \\ \mathfrak{A} \models \phi[s'], & \end{array}$$

as s and s' agree on the free variables of ϕ.

Sub-subcase 2: If $\phi :\equiv \forall y\psi$, y is not x, and x *is* free in ψ, then as t is substitutable for x in ϕ (we had to use that assumption somewhere, didn't we?), we know that y does not occur in t and t is substitutable for x in ψ. Then we have

$$\begin{array}{ll} \mathfrak{A} \models \phi_t^x[s] & \text{iff} \\ \mathfrak{A} \models (\forall y)(\psi_t^x)[s] & \text{iff} \\ \mathfrak{A} \models (\psi_t^x)[s[y|a]] & \text{for every } a \in A. \end{array}$$

But we also know that

$$\begin{array}{ll} \mathfrak{A} \models \phi[s'] & \text{iff} \\ \mathfrak{A} \models (\forall y)(\psi)[s'] & \text{iff} \\ \mathfrak{A} \models \psi[s'[y|a]] & \text{for every } a \in A. \end{array}$$

But since x is not y, we know that for any $a \in A$, $s'[y|a] = s[y|a][x|\overline{s}(t)]$, so by the inductive hypothesis (notice that t is substitutable for x in ψ) we have

$$\mathfrak{A} \models (\psi_t^x)[s[y|a]] \text{ iff } \mathfrak{A} \models \psi[s'[y|a]].$$

So $\mathfrak{A} \models \phi_t^x[s]$ if and only if $\mathfrak{A} \models \phi[s']$, as needed. □

2.7 Properties of Our Deductive System

Having gone through all the trouble of setting out our deductive system, we will now prove a few things both in and about that system. First, we will show that we can prove, in our deductive system, that equality is an equivalence relation.

Theorem 2.7.1.

1. $\vdash x = x$.
2. $\vdash x = y \to y = x$.
3. $\vdash (x = y \wedge y = z) \to x = z$.

Proof. We show that we can find deductions establishing that $=$ is reflexive, symmetric, and transitive in turn.

1. This is a logical axiom of type (E1).
2. Here is the needed deduction. Notice that the notations off to the right are listed only as an aid to the reader.

$$\begin{array}{ll} [x = y \wedge x = x] \to [x = x \to y = x] & \text{(E3)} \\ x = x & \text{(E1)} \\ x = y \to y = x. & \text{(PC)} \end{array}$$

3. Again, we present a deduction:

$$\begin{array}{ll} [x = x \wedge y = z] \to [x = y \to x = z] & \text{(E3)} \\ x = x & \text{(E1)} \\ (x = y \wedge y = z) \to x = z. & \text{(PC)} \quad \square \end{array}$$

Chaff: Notice that we have done a bit more than prove that equality is an equivalence relation. (Heck, you've known *that* since fourth grade.) Rather, we've shown that our deductive system, with the axioms and rules of inference that have been outlined in this chapter, is powerful enough to *prove* that equality is an equivalence relation. There will be a fair bit of "our deductive system is strong enough to do such-and-such" in the pages to come.

We now prove some general properties of our deductive system. We start off with a lemma that seems somewhat problematical, but it will help us to think a little more carefully about what our deductions do for us.

Lemma 2.7.2. *$\Sigma \vdash \theta$ if and only if $\Sigma \vdash \forall x \theta$.*

Proof. First, suppose that $\Sigma \vdash \theta$. Here is a deduction from Σ of $\forall x\theta$:

$$\begin{array}{ll}
\vdots & \text{Deduction of } \theta \\
\theta & \\
\big[(\forall y(y = y)) \vee \neg(\forall y(y = y))\big] \to \theta & \text{(PC)} \\
\big[(\forall y(y = y)) \vee \neg(\forall y(y = y))\big] \to (\forall x\theta) & \text{(QR)} \\
\forall x\theta. & \text{(PC)}
\end{array}$$

There are a couple of things to point out about this proof. The first use of (PC) is justified by the fact that if θ is true, then (anything $\to \theta$) is also true. The second use of (PC) depends on the fact that $\big[(\forall y(y = y)) \vee \neg(\forall y(y = y))\big]$ is a tautology, and thus $\forall x\theta$ is a propositional consequence of the implication. As for the (QR) step of the deduction, notice that the variable x is not free in the sentence $\big[(\forall y(y = y)) \vee \neg(\forall y(y = y))\big]$, making the use of the quantifier rule legitimate.

Now, suppose that $\Sigma \vdash \forall x\theta$. Here is a deduction from Σ of θ (recall that θ^x_x is θ):

$$\begin{array}{ll}
\vdots & \text{Deduction of } \forall x\theta \\
\forall x\theta & \\
\forall x\theta \to \theta^x_x & \text{(Q1)} \\
\theta^x_x. & \text{(PC)}
\end{array}$$

Thus $\Sigma \vdash \theta$ if and only if $\Sigma \vdash \forall x\theta$. □

Here is an example to show how strange this lemma might seem. Suppose that Σ consists of the single formula $x = \overline{5}$. Then certainly $\Sigma \vdash x = \overline{5}$, and so, by the lemma, $\Sigma \vdash (\forall x)(x = \overline{5})$. You might be tempted to say that by assuming x was equal to five, we have proved that *everything* is equal to five. But that is not quite what is going on. If $x = \overline{5}$ is true in a model $\mathfrak{A}$, that means that $\mathfrak{A} \models x = \overline{5}[s]$ for every assignment function s. And since for every $a \in A$, there is an assignment function s such that $s(x) = a$, it must be true that every element of A is equal to 5, so the universe A has only one element, and everything *is* equal to 5. So our deduction of $(\forall x)(x = \overline{5})$ has preserved truth, but our assumption was much stronger than it appeared at first glance. And the moral of our story is: For a formula to be true in a structure, it must be satisfied in that structure with *every* assignment function.

Lemma 2.7.3. *Suppose that $\Sigma \vdash \theta$. Then if Σ' is formed by taking any $\sigma \in \Sigma$ and adding or deleting a universal quantifier whose scope is the entire formula, $\Sigma' \vdash \theta$.*

Proof. This follows immediately from Lemma 2.7.2. Suppose that $\forall x\sigma$ is in Σ'. By the preceding, $\Sigma' \vdash \sigma$. Then, given a deduction from Σ of θ, to

produce a deduction from Σ' of θ, first write down a deduction from Σ' of σ, and then copy your deduction from Σ of θ. Having already established σ, this deduction will be a valid deduction from Σ'.

The proof in the case that $\forall x\sigma$ is an element of Σ and it is replaced by σ in Σ' is analogous. □

Notice that one consequence of this lemma is the fact that if we know $\Sigma \vdash \theta$, we can assume (if we like) that every element of Σ is a sentence: By quoting Lemma 2.7.3 several times, we can replace each $\sigma \in \Sigma$ with its universal closure.

Now we will show that in at least some sense, the system of deductions that we have developed mirrors the process that mathematicians use to prove theorems. Suppose you were asked to prove the theorem: *If A is a square, then A is a rectangle.* A perfectly reasonable way to attack this theorem would be to assume that A is a square, and using that assumption, prove that A is a rectangle. But notice that you have not been asked to prove that A is a rectangle. You were asked to prove an implication! The Deduction Theorem says that there is a deduction of ϕ from the assumption θ if and only if there is a deduction of the implication $\theta \to \phi$. (A bit of notation: Rather than writing the formally correct $\Sigma \cup \{\theta\} \vdash \phi$, we shall omit the braces and write $\Sigma \cup \theta \vdash \phi$.)

Theorem 2.7.4 (The Deduction Theorem). *Suppose that θ is a sentence and Σ is a set of formulas. Then $\Sigma \cup \theta \vdash \phi$ if and only if $\Sigma \vdash (\theta \to \phi)$.*

Proof. First, suppose that $\Sigma \vdash (\theta \to \phi)$. Then, as the same deduction would show that $\Sigma \cup \theta \vdash (\theta \to \phi)$, and as $\Sigma \cup \theta \vdash \theta$ by a one-line deduction, and as ϕ is a propositional consequence of θ and $(\theta \to \phi)$, we know that $\Sigma \cup \theta \vdash \phi$.

For the more difficult direction we will make use of Proposition 2.2.4. Suppose that $C = \{\phi \mid \Sigma \vdash (\theta \to \phi)\}$. If we show that C contains $\Sigma \cup \theta$, C contains all the axioms of Λ, and C is closed under the rules of inference as noted in Proposition 2.2.4, then by that proposition we will know that $\{\phi \mid \Sigma \cup \theta \vdash \phi\} \subseteq C$. In other words, we will know that if $\Sigma \cup \theta \vdash \phi$, then $\Sigma \vdash (\theta \to \phi)$, which is what we need to show.

So it remains to prove that C has the properties listed in the preceding paragraph.

1. $\Sigma \subseteq C$: If $\sigma \in \Sigma$, then $\Sigma \vdash \sigma$. But then $\Sigma \vdash (\theta \to \sigma)$, as this is a propositional consequence of σ.

2. $\theta \in C$: $\Sigma \vdash \theta \to \theta$, as this is a tautology.

3. $\Lambda \subseteq C$: This is identical to (1).

4. C is closed under the rules:

(a) *Rule (PC):* Suppose that $\gamma_1, \gamma_2, \ldots, \gamma_n$ are all elements of C and ϕ is a propositional consequence of $\{\gamma_1, \gamma_2, \ldots, \gamma_n\}$. We must show that $\phi \in C$. By assumption, $\Sigma \vdash (\theta \to \gamma_1)$, $\Sigma \vdash (\theta \to \gamma_2)$, $\ldots$, $\Sigma \vdash (\theta \to \gamma_n)$. But then as $(\theta \to \phi)$ is a propositional consequence of the set

$$\{(\theta \to \gamma_1), (\theta \to \gamma_2), \ldots, (\theta \to \gamma_n)\},$$

we have that $\Sigma \vdash (\theta \to \phi)$. In other words, $\phi \in C$, as needed.

(b) *Quantifier Rules:* Suppose that $\psi \to \phi$ is in C and x is not free in ψ. We want to show that $(\psi \to \forall x\phi)$ is an element of C. In other words, we have to show that

$$\Sigma \vdash \big[\theta \to (\psi \to \forall x\phi)\big].$$

By assumption we have

$\Sigma \vdash \big[\theta \to (\psi \to \phi)\big]$	$\psi \to \phi$ is in C
$\Sigma \vdash (\theta \wedge \psi) \to \phi$	propositional consequence
$\Sigma \vdash (\theta \wedge \psi) \to \forall x\phi$	rule (QR)
$\Sigma \vdash \big[\theta \to (\psi \to \forall x\phi)\big]$	propositional consequence

Notice that our use of rule (QR) is legitimate since we know that θ is a sentence, so x is not free in θ. But the last line of our argument says that $(\psi \to \forall x\phi) \in C$, which is what we needed to show.

The other quantifier rule, dealing with the existential quantifier, is proved similarly.

So we have shown that C contains θ, all the elements of Σ and Λ, and C is closed under the rules. This finishes the proof of the Deduction Theorem. □

2.7.1 Exercises

1. Lemma 2.7.2 tells us that $\Sigma \vdash \theta$ if and only if $\Sigma \vdash \forall x\theta$. What happens if we replace the universal quantifier by an existential quantifier? So suppose that $\Sigma \vdash \theta$. Must $\Sigma \vdash \exists x\theta$? Now assume that $\Sigma \vdash \exists x\theta$. Does Σ necessarily prove θ?

2. Finish the proof of Lemma 2.7.3 by considering the case when $\forall x\sigma$ is an element of Σ and is replaced by σ in Σ'.

3. Many authors demand that axioms be sentences rather than formulas. Explain how Lemma 2.7.2 implies that we could replace all of our axioms by their universal closures without changing the strength of our deductive system.

4. Suppose that η is a sentence. Prove that $\Sigma \vdash \eta$ if and only if $\Sigma \cup (\neg\eta) \vdash \big[(\forall x)x = x\big] \wedge \neg\big[(\forall x)x = x\big]$. Notice that this exercise tells us that our deductive system allows us to do proofs by contradiction.

5. Suppose that P is a unary relation symbol and show that

$$\vdash \big[(\forall x)P(x)\big] \to \big[(\exists x)P(x)\big].$$

[*Suggestion:* Proof by contradiction (see Exercise 4) works nicely here.]

6. If P is a binary relation symbol, show that

$$(\forall x)(\forall y)P(x, y) \vdash (\forall y)(\forall z)P(z, y).$$

7. Let P and Q be unary relation symbols, and show that

$$\vdash [(\forall x)(P(x)) \wedge (\forall x)(Q(x))] \to (\forall x)\,[P(x) \wedge Q(x)]\,.$$

2.8 Nonlogical Axioms

When we are trying to prove theorems in mathematics, there are almost always additional axioms, beyond the set of logical axioms Λ, that we use. If we are trying to prove a theorem about vector spaces, the axioms of vector spaces come in mighty handy. If we are proving theorems in a real analysis course, we need to have axioms about the structure of the real numbers. These additional axioms are sometimes explicitly stated and sometimes they are blanket assumptions that are made without being stated, but they are almost always there. In this section we give a couple of examples of sets of nonlogical axioms that we might use in writing deductions.

Example 2.8.1. For many of us, the first explicit set of nonlogical axioms that we see is in a course on linear algebra. To work those axioms out explicitly, let us fix the language $\mathcal{L}$ as consisting of one binary function symbol, $\oplus$, and infinitely many unary function symbols, $c\cdot$, one for each real number c. (Yes, that symbol is "c-dot.") These function symbols will be used to represent the functions of scalar multiplication. We will also have one constant symbol, 0, to represent the zero vector of the vector space. Here, then, is one way to list the nonlogical axioms of a vector space:

1. $(\forall x)(\forall y)x \oplus y = y \oplus x$ (vector addition is commutative).
2. $(\forall x)(\forall y)(\forall z)x \oplus (y \oplus z) = (x \oplus y) \oplus z$ (vector addition is associative).
3. $(\forall x)x \oplus 0 = x$.
4. $(\forall x)(\exists y)x \oplus y = 0$.
5. $(\forall x)1 \cdot x = x$.

6. $(\forall x)(c_1c_2) \cdot x = c_1 \cdot (c_2 \cdot x)$.
7. $(\forall x)(\forall y)c \cdot (x \oplus y) = c \cdot x \oplus c \cdot y$.
8. $(\forall x)(c_1 + c_2) \cdot x = c_1 \cdot x \oplus c_2 \cdot x$.

Notice a couple of things here: There are infinitely many axioms listed, as the last three axioms are really axiom schemas, consisting of one axiom for each choice of c, c_1, and c_2. An axiom schema is a template, saying that a formula is in the axiom set if it is of a certain form. Also notice that I've cheated in using the addition sign to stand for addition and juxtaposition to stand for multiplication of real numbers since the language $\mathcal{L}$ does not allow that sort of thing. See Exercise 1.

Example 2.8.2. We will write out the axioms for a dense linear order without endpoints. Our language consists of a single binary relation symbol, $<$. Our nonlogical axioms are:

1. $(\forall x)(\forall y)(x < y \vee x = y \vee y < x)$.
2. $(\forall x)(\forall y)[x = y \rightarrow \neg x < y]$.
3. $(\forall x)(\forall y)(\forall z)[(x < y \wedge y < z) \rightarrow x < z]$.
4. $(\forall x)(\forall y)\big[x < y \rightarrow \big((\exists z)(x < z \wedge z < y)\big)\big]$.
5. $(\forall x)(\exists y)(\exists z)(y < x \wedge x < z)$.

The first three axioms guarantee that the relation denoted by $<$ is a linear order, the fourth axiom states that the relation is dense, and the final axiom ensures that there is no smallest element and no greatest element.

Notice that in both of our examples, the axiom set involved is decidable: Given a formula ϕ that is alleged to be either an axiom for vector spaces or an axiom for dense linear orders without endpoints, we could decide whether or not the formula was, in fact, such an axiom. And furthermore, we could write a computer program that could decide the issue for us.

Example 2.8.3. It is time to introduce a collection of nonlogical axioms that will be vitally important to us for the rest of the book. We work in the language of number theory,

$$\mathcal{L}_{NT} = \{0, S, +, \cdot, E, <\}.$$

The set of axioms we will call N is a minimal set of assumptions to describe a bare-bones version of the usual operations on the set of natural numbers. Just how weak these axioms are will be discussed in the next chapter. These axioms will, however, be important to us in Chapters 4, 5, and 6 precisely because they are so weak.

The Axioms of N

1. $(\forall x)\neg Sx = 0$.
2. $(\forall x)(\forall y)\big[Sx = Sy \to x = y\big]$.
3. $(\forall x)x + 0 = x$.
4. $(\forall x)(\forall y)x + Sy = S(x + y)$.
5. $(\forall x)x \cdot 0 = 0$.
6. $(\forall x)(\forall y)x \cdot Sy = (x \cdot y) + x$.
7. $(\forall x)xE0 = S0$.
8. $(\forall x)(\forall y)xE(Sy) = (xEy) \cdot x$.
9. $(\forall x)\neg x < 0$.
10. $(\forall x)(\forall y)\big[x < Sy \leftrightarrow (x < y \vee x = y)\big]$.
11. $(\forall x)(\forall y)\big[(x < y) \vee (x = y) \vee (y < x)\big]$.

Although we have just claimed that N is a weak set of axioms, let us show that N is strong enough to prove some of the basic facts about the relations and functions on the natural numbers. For the following discussion, if a is a natural number, let $\overline{a}$ be the $\mathcal{L}_{NT}$-term $\underbrace{SS\cdots S}_{a S\text{'s}}0$. So $\overline{a}$ is the canonical term of the language that is intended to refer to the natural number a.

Lemma 2.8.4. *For natural numbers a and b:*

1. *If $a = b$, then $N \vdash \overline{a} = \overline{b}$.*
2. *If $a \neq b$, then $N \vdash \overline{a} \neq \overline{b}$.*
3. *If $a < b$, then $N \vdash \overline{a} < \overline{b}$.*
4. *If $a \not< b$, then $N \vdash \overline{a} \not< \overline{b}$.*
5. $N \vdash \overline{a} + \overline{b} = \overline{a + b}$
6. $N \vdash \overline{a} \cdot \overline{b} = \overline{a \cdot b}$
7. $N \vdash \overline{a}E\overline{b} = \overline{a^b}$

Proof. Let us begin with (1), and let us work rather carefully. Notice that the theorem is saying that if the *number* a is equal to the *number* b, then there is a deduction from the axioms in N of the formula

$$\underbrace{SS\cdots S}_{aS's}0 = \underbrace{SS\cdots S}_{bS's}0.$$

We work by induction on a (and b, since $a = b$). So, first assume that $a = b = 0$. Here is the needed deduction in N:

$\vdots$ Deduction of $(\forall x)x = x$ (see Lemma 2.7.2)
$(\forall x)x = x$
$(\forall x)x = x \to 0 = 0$ (Q1)
$0 = 0.$ (PC)

Now, what if $a = b$ and a and b are greater than 0? Then certainly $a-1$ and $b-1$ are equal, and by the inductive hypothesis there is a deduction of $\underbrace{SS\cdots S}_{a-1 S's}0 = \underbrace{SS\cdots S}_{b-1 S's}0$. If we follow that deduction with a use of axiom (E2): $x = y \to Sx = Sy$, and then (PC) gives us $\underbrace{SS\cdots S}_{aS's}0 = \underbrace{SS\cdots S}_{bS's}0$, as needed. Write out the details of the end of this deduction. It is a little trickier than we have made it sound when you actually have to use (Q1) to do the substitution. This finishes the inductive step of the proof, so (1) is established. (Alternatively, you can establish (1) using the axiom (E1) and several applications of (E2), but we thought you should see the inductive proof for practice.)

Looking at (2), suppose that $a \neq b$. If one of a or b is 0, then $\neg\overline{a} = \overline{b}$ follows quickly from Axiom N1 and the fact that N proves that $=$ is an equivalence relation. If neither a nor b is 0, we proceed by induction on the smaller of a, b. Since $a - 1 \neq b - 1$, by the inductive hypothesis, $N \vdash \neg\overline{a-1} = \overline{b-1}$. Then by Axiom N2, $N \vdash \neg S(\overline{a-1}) = S(\overline{b-1})$. In other words, $N \vdash \neg\overline{a} = \overline{b}$, as $S(\overline{a-1})$ is typographically equivalent to $\overline{a}$ and $S(\overline{b-1})$ is typographically equivalent to $\overline{b}$.

For (3), we use induction on b. As $a < b$, we know that $b \neq 0$ and we know that $a < b - 1$ or $a = b - 1$. So either

$$N \vdash \overline{a} < \overline{b-1} \text{ (by the inductive hypothesis)}$$

or

$$N \vdash \overline{a} = \overline{b-1} \text{ (by (1))}.$$

So

$$N \vdash (\overline{a} < \overline{b-1} \vee \overline{a} = \overline{b-1}).$$

But then by Axiom N10, $N \vdash \overline{a} < S(\overline{b-1})$, which is exactly the same as $N \vdash \overline{a} < \overline{b}$.

We will now discuss (5), leaving (4), (6), and (7) to the exercises. We prove (5) by induction on b. If $b = 0$, then $\overline{a+b} :\equiv \overline{a+0} :\equiv \overline{a}$. So Axiom N3 tells us that $N \vdash \overline{a} + \overline{b} = \overline{a}$.

For the inductive step, if $b = c + 1$, then $\overline{a} + \overline{b} :\equiv \overline{a} + S(\overline{c})$. So Axiom N4 tells us that

$$N \vdash \overline{a} + \overline{b} = S(\overline{a} + \overline{c}).$$

Since $N \vdash \overline{a} + \overline{c} = \overline{a + c}$ by the inductive hypothesis, the equality axioms tell us that $N \vdash S(\overline{a} + \overline{c}) = S(\overline{a + c})$. But $S(\overline{a + c})$ *is* $\overline{a + c + 1}$, which is $\overline{a + b}$. Since we know (by Theorem 2.7.1) that $N \vdash$ "equality is transitive," $N \vdash \overline{a} + \overline{b} = \overline{a + b}$. □

2.8.1 Exercises

1. This problem is in the setting of Example 2.8.1. Exactly one of the following two statements is in the collection of nonlogical axioms of that example. Figure out which one it is, and why.

   - $(\forall x)(17 + 42) \cdot x = 17 \cdot x \oplus 42 \cdot x$.
   - $(\forall x)59 \cdot x = 17 \cdot x \oplus 42 \cdot x$.

   Now fix up the presentation of the axioms for a vector space. You may need to redefine the language, or you may be able to take what is presented in Example 2.8.1 and fix it up.

2. For each of the following structures, decide whether or not it satisfies all of the axioms of Example 2.8.2. If the structure is *not* a dense linear order without endpoints, point out which of the axioms the structure fails to satisfy.

   (a) The structure $(\mathbb{N}, <)$, the natural numbers with the usual less than relation

   (b) The structure $(\mathbb{Z}, <)$, the integers with the usual less than relation

   (c) The structure $(\mathbb{Q}, <)$, the set of rational numbers with the usual less than relation

   (d) The structure $(\mathbb{R}, <)$, the real numbers with the usual less than relation

   (e) The structure $(\mathbb{C}, <)$, the complex numbers with the relation $<$ defined by:

   $$a + bi < c + di \text{ if and only if } (a^2 + b^2) < (c^2 + d^2).$$

3. Write out the axioms for group theory. If you do not know the axioms of group theory, go to the library and check out any book with the phrase "abstract algebra," "modern algebra," or "group theory" in the title. Then check the index under "group." Specify your language carefully and then writing out the axioms should be easy.

4. In this exercise you are asked to write up some of the axioms of Zermelo–Fraenkel set theory, also known as ZF. The language of set theory consists of a single binary relation symbol, $\in$, that is intended to represent the relation "is an element of." So the formula $x \in y$ will usually be interpreted as meaning that the set x is an element of the set y. Here are

English versions of some of the axioms of ZF. Write them up formally as sentences in the language of set theory.

The Axiom of Extensionality: Two sets are equal if and only if they have the same elements.

The Null Set Axiom: There is a set with no elements.

The Pair Set Axiom: If a and b are sets, then there is a set whose only elements are a and b.

The Axiom of Union: If a is a set, then there is a set consisting of exactly the elements of the elements of a. [*Query:* Can you figure out why this is called the axiom of union? Write up an example, where a is a set of three sets and each of those three sets has two elements. What does the set whose existence is guaranteed by this axiom look like?]

The Power Set Axiom: If a is a set, then there is a set consisting of all of the subsets of a. [*Suggestion:* For this axiom it might be nice to define $\subseteq$ by saying that $x \subseteq y$ is shorthand for (some nice formula with x and y free in the language of set theory).]

5. Complete the proof of Lemma 2.8.4.

6. Lemma 2.8.4(2) states that there is a deduction in N of the sentence $\neg(= S0SS0)$. Find a deduction in N of this sentence.

7. This problem is just to give you a hint of how little we can prove using the axiom system N. Suppose that we wanted to prove that $N \nvdash \neg x < x$. It makes sense (and is a consequence of the Soundness Theorem, Theorem 2.5.3) that one way to go about this would be to construct an $\mathcal{L}_{NT}$-structure $\mathfrak{A}$ in which all the axioms of N are true but $(\forall x)\neg x < x$ is not true. Do so. We would suggest that you take as your universe the set
$$A = \{0, 1, 2, 3, \dots\} \cup \{a\},$$
where a is the letter a and not a natural number. You need to define the functions $S^{\mathfrak{A}}, +^{\mathfrak{A}}$, etc., and the relation $<^{\mathfrak{A}}$. Don't do anything too strange for the natural numbers, but make sure that $a <^{\mathfrak{A}} a$. Check that the axioms of N are true in the structure $\mathfrak{A}$, and you're finished!

8. Using more or less the same technique as in Exercise 7, show that N does not prove that addition is commutative.

2.9 Summing Up, Looking Ahead

In these first two chapters we have developed a vocabulary for talking about mathematical structures, mathematical languages, and deductions. Chap-

ter 2 has focused on deductions, which are supposed to be the formal equivalents of the mathematical proofs that you have seen for many years. We have seen some results, such as the Deduction Theorem, which indicate that deductions behave like proofs behave. The Soundness Theorem shows that deductions preserve truth, which gives us some comfort as we try to justify in our minds why proofs preserve truth.

As you look at the statement of the Soundness Theorem, you can see that it is explicitly trying to relate the syntactical notion of deducibility ($\vdash$) with the semantical notion of logical implication ($\models$). The first major result of Chapter 3, the Completeness Theorem, will also relate these two notions and will in fact show that they are equivalent. Then the Compactness Theorem (which is really a quite trivial consequence of the Completeness Theorem) will be used to construct some mathematical models with some very interesting properties.

Chapter 3

Completeness and Compactness

3.1 Naïvely

We are at a point in our explorations where we have established a particular deductive system, consisting of the logical axioms and rules of inference that we set out in the last chapter. The Soundness Theorem showed that our deductive system preserves truth, in the sense that if there is a deduction of ϕ from Σ, then ϕ is true in any model of Σ. The Completeness Theorem, the first major result of this chapter, gives us the converse to the Soundness Theorem. So, when the two results are combined, we will have this equivalence:

$$\Sigma \models \phi \text{ if and only if } \Sigma \vdash \phi.$$

We have already made a big point of the fact that we would like to be sure that if our deductive system allows us to prove a statement, we would like that statement to be true. Certainly, the content of the Soundness Theorem is exactly that. If $\vdash \phi$, if there is a deduction of ϕ from only the logical axioms without any additional assumptions, then we know that $\models \phi$, so ϕ is true in every structure with every assignment function. To the extent that the informal mathematical practice of everyday proofs is modeled by our formal system of deduction, we can be sure that the things that we prove mathematically are true.

If life were peaches and cream, we would also like to know that we can prove anything that is true. The Completeness Theorem is the result that asserts that our deductive system *is* that strong. So you would be tempted to conclude that, for example, we are able to prove any statement of first-order logic that is a true statement about the natural numbers.

Unfortunately, this conclusion is based upon a misreading of the state-

ment of the Completeness Theorem. What we will prove is that our deductive system is complete, in the sense of this definition:

Definition 3.1.1. A deductive system consisting of a collection of logical axioms Λ and a collection of rules of inference is said to be **complete** if for every set of nonlogical axioms Σ and every $\mathcal{L}$-formula ϕ,

$$\text{If } \Sigma \models \phi, \text{ then } \Sigma \vdash \phi.$$

What this says is that if ϕ is an $\mathcal{L}$-formula that is true in *every* model of Σ, then there will be a deduction from Σ of ϕ. So our ability to prove ϕ depends on ϕ being true in every model of Σ. Thus if we want to be able to use Σ to prove every true statement about the natural numbers, we have to be able to find a set of non-logical axioms Σ such that $\Sigma \models \phi$ if and only if ϕ is a true statement about the natural numbers. We will have much more to say about that problem in Chapters 4, 5, 6, and 7.

The second part of the chapter concerns the Compactness Theorem and the Löwenheim–Skolem Theorems. We will use these results to investigate various types of mathematical structures, including structures that are quite surprising.

In some sense, we have spent a lot of time in the first couple of chapters of this book developing a lot of vocabulary and establishing some basic results. Now we will roll up our sleeves and get a couple of worthwhile theorems. It is time to start showing some of the beauty and the power, as well as the limitations, of first-order logic.

3.2 Completeness

Let us fix a collection of nonlogical axioms, Σ. Our goal in this section is to show that for any formula ϕ, if $\Sigma \models \phi$, then $\Sigma \vdash \phi$. In some sense, this is the only possible interpretation of the phrase "you can prove anything that is true," if you are discussing the adequacy of the deductive system. To say that ϕ is true whenever Σ is a collection of true axioms is precisely to say that Σ logically implies ϕ. Thus, the Completeness Theorem will say that whenever ϕ is logically implied by Σ, there is a deduction from Σ of ϕ. So the Completeness Theorem is the converse of the Soundness Theorem.

We have to begin with a short discussion of consistency.

Definition 3.2.1. Let Σ be a set of $\mathcal{L}$-formulas. We will say that Σ is **inconsistent** if there is a deduction from Σ of $\big[(\forall x)x = x\big] \wedge \neg\big[(\forall x)x = x\big]$. We say that Σ is **consistent** if it is not inconsistent.

So Σ is inconsistent if Σ proves a contradiction. Exercise 1 asks you to show that if Σ is inconsistent, then there is a deduction from Σ of every $\mathcal{L}$-formula. For notational convenience, let us agree to use the symbol $\perp$ (read

"false" or "eet") for the contradictory sentence $\big[(\forall x)x = x\big] \wedge \neg\big[(\forall x)x = x\big]$. All you will have to remember is that $\bot$ is a sentence that is in every language and is true in no structure.

Theorem 3.2.2 (Completeness Theorem). *Suppose that Σ is a set of $\mathcal{L}$-formulas and ϕ is an $\mathcal{L}$-formula. If $\Sigma \models \phi$, then $\Sigma \vdash \phi$.*

Proof.

> *Chaff:* This theorem was established in 1929 by the Austrian mathematician Kurt Gödel, in his PhD dissertation. If you haven't picked it up already, you should know that the work of Gödel is central to the development of logic in the twentieth century. He is responsible for most of the major results that we will state in the rest of the book: The Completeness Theorem, the Compactness Theorem, and the two Incompleteness Theorems. Gödel was an absolutely brilliant man, with a complex and troubled personality. A wonderful and engaging biography of Gödel is [Dawson 97]. The first volume of Gödel's collected works, [Gödel–Works], also includes a biography and introductory comments about his papers that can help your understanding of this wonderful mathematics.
>
> The proof we present of the Completeness Theorem is based on work of Leon Henkin. The idea of Henkin's proof is brilliant, but the details take some time to work through. Just to warn you, this proof doesn't end until page 84.

Before we get involved in the details, let us look at a rough outline of how the argument proceeds. There are a few simplifications and one or two outright lies in the outline, but we will straighten everything out as we work out the proof.

Outline of the Proof

There will be a **preliminary argument** that will show that it is sufficient to prove that if Σ is a consistent set of sentences, then Σ has a model. Then we will proceed to assume that we are given such a set of sentences, and we will construct a model for Σ.

The construction of the model will proceed in several steps, but the central idea was introduced in Example 1.6.4. The elements of the model will be variable-free terms of a language. We will construct this model so that the formulas that will be true in the model are precisely the formulas that are in a certain set of formulas, which we will call Σ'. We will make sure that $\Sigma \subseteq \Sigma'$, so all of the formulas of Σ will be true in this constructed model. In other words, we will have constructed a model of Σ.

To make the construction work we will take our given set of $\mathcal{L}$-sentences Σ and extend it to a bigger set of sentences Σ' in a bigger language $\mathcal{L}'$. We do this extension in two steps. First, we will add in some new axioms, called Henkin Axioms, to get a collection $\hat{\Sigma}$. Then we will extend $\hat{\Sigma}$ to Σ' in such a way that:

1. Σ' is consistent.

2. For every $\mathcal{L}'$-sentence θ, either $\theta \in \Sigma'$ or $(\neg\theta) \in \Sigma'$.

Thus we will say that Σ' is a maximal consistent extension of Σ, where *maximal* means that it is impossible to add any sentences to Σ' without making Σ' inconsistent.

Now there are two possible sources of problems in this expansion of Σ to Σ'. The first is that we will **change languages from $\mathcal{L}$ to $\mathcal{L}'$**, where $\mathcal{L} \subseteq \mathcal{L}'$. It is conceivable that Σ will not be consistent when viewed as a set of $\mathcal{L}'$-sentences, even though Σ is consistent when viewed as a set of $\mathcal{L}$-sentences. The reason that this might happen is that there are more $\mathcal{L}'$-deductions than there are $\mathcal{L}$-deductions, and one of these new deductions just might happen to be a deduction of $\perp$. Fortunately, Lemma 3.2.3 will show us that this does not happen, so Σ is consistent as a set of $\mathcal{L}'$-sentences. The other possible problem is in our two **extensions of** Σ, first to $\hat{\Sigma}$ and then to Σ'. It certainly might happen that we could add a sentence to Σ in such a way as to make Σ' inconsistent. But Lemma 3.2.4 and Exercise 4 will prove that Σ' is still consistent.

Once we have our maximal consistent set of sentences Σ', we will **construct a model** $\mathfrak{A}$ and prove that the sentences of $\mathcal{L}'$ that are in Σ' are precisely the sentences that are true in $\mathfrak{A}$. Thus, $\mathfrak{A}$ will be a model of Σ', and as $\Sigma \subseteq \Sigma'$, $\mathfrak{A}$ will be a model of Σ, as well.

This looks daunting, but if we keep our wits about us and do things one step at a time, it will all come together at the end.

Preliminary Argument

So let us fix our setting for the rest of this proof. We are working in a language $\mathcal{L}$. For the purposes of this proof, we assume that the language is countable, which means that the formulas of $\mathcal{L}$ can be written in an infinite list $\alpha_1, \alpha_2, \ldots, \alpha_n, \ldots$. (An outline of the changes in the proof necessary for the case when $\mathcal{L}$ is not countable can be found in Exercise 6.)

We are given a set of formulas Σ, and we are assuming that $\Sigma \models \phi$. We have to prove that $\Sigma \vdash \phi$.

Note that we can assume that ϕ is a sentence: By Lemma 2.7.2, $\Sigma \vdash \phi$ if and only if there is a deduction from Σ of the universal closure of ϕ. Also, by the comments following Lemma 2.7.3, we can also assume that every element of Σ is a sentence. So, now all(!) we have to do is prove that if Σ is a set of *sentences* and ϕ is a *sentence* and if $\Sigma \models \phi$, then $\Sigma \vdash \phi$.

Now we claim that it suffices to prove the case where ϕ is the sentence $\bot$. For suppose we know that if $\Sigma \models \bot$, then $\Sigma \vdash \bot$, and suppose we are given a sentence ϕ such that $\Sigma \models \phi$. Then $\Sigma \cup (\neg\phi) \models \bot$, as there are no models of $\Sigma \cup (\neg\phi)$, so $\Sigma \cup (\neg\phi) \vdash \bot$. This tells us, by Exercise 4 in Section 2.7.1, that $\Sigma \vdash \phi$, as needed.

So we have reduced what we need to do to proving that if $\Sigma \models \bot$, then $\Sigma \vdash \bot$, for Σ a set of $\mathcal{L}$-sentences. This is equivalent to saying that if there is no model of Σ, then $\Sigma \vdash \bot$. We will work with the contrapositive: If $\Sigma \nvdash \bot$, then there is a model of Σ. In other words, we will prove:

If Σ is a consistent set of sentences, then there is a model of Σ.

This ends the preliminary argument that was promised in the outline of the proof. Now, we will assume that Σ is a consistent set of $\mathcal{L}$-sentences and go about the task of constructing a model of Σ.

Changing the Language from $\mathcal{L}$ to $\mathcal{L}_1$

The model of Σ that we will construct will be a model whose elements are variable-free terms of a language. This might lead to problems. For example, suppose that $\mathcal{L}$ contains no constant symbols. Then there will be no variable-free terms of $\mathcal{L}$. Or, perhaps $\mathcal{L}$ has exactly one constant symbol c, no function symbols, one unary relation P, and

$$\Sigma = \{\exists x P(x), \neg P(c)\}.$$

Here Σ is consistent, but no structure whose universe is $\{c\}$ (c is the only variable-free term of $\mathcal{L}$) can be a model of Σ. So we have to expand our language to give us enough constant symbols to build our model.

So let $\mathcal{L}_0 = \mathcal{L}$, and define

$$\mathcal{L}_1 = \mathcal{L}_0 \cup \{c_1, c_2, \ldots, c_n, \ldots\},$$

where the c_i's are new constant symbols.

> *Chaff:* Did you notice that when we were defining $\mathcal{L}_1$ we took something we already knew about, $\mathcal{L}$, and gave it a new name, $\mathcal{L}_0$? When you are reading mathematics and something like that happens, it is almost always a clue that whatever happens next is going to be iterated, in this case to build $\mathcal{L}_2, \mathcal{L}_3$, and so on. In those literature courses we took, they called that foreshadowing.

We say (for the obvious reason) that $\mathcal{L}_1$ is an **extension by constants** of $\mathcal{L}_0$. As mentioned in the outline, it is not immediately clear that Σ remains consistent when viewed as a collection of $\mathcal{L}_1$-sentences rather than $\mathcal{L}$-sentences. The following lemma, the proof of which is delayed until page 84, shows that Σ remains consistent.

Lemma 3.2.3. *If Σ is a consistent set of $\mathcal{L}$-sentences and $\mathcal{L}_1$ is an extension by constants of $\mathcal{L}$, then Σ is consistent when viewed as a set of $\mathcal{L}_1$-sentences.*

The constants that we have added to form $\mathcal{L}_1$ are called **Henkin constants**, and they serve a particular purpose. They will be the witnesses that allow us to ensure that any time Σ claims $\exists x \phi(x)$, then in our constructed model $\mathfrak{A}$, there will be an element (which will be one of these constants c) such that $\mathfrak{A} \models \phi(c)$.

> *Chaff:* Recall that the notation $\exists x \phi(x)$ implies that ϕ is a formula with x as the only free variable. Then $\phi(c)$ is the result of replacing the free occurrences of x with the constant symbol c. Thus $\phi(c)$ is ϕ^x_c.

The next step in our construction makes sure that the Henkin constants will be the witnesses for the existential sentences in Σ.

Extending Σ to Include Henkin Axioms

Consider the collection of sentences of the form $\exists x \theta$ in the language $\mathcal{L}_0$. As the language $\mathcal{L}_0$ is countable, the collection of $\mathcal{L}_0$-sentences is countable, so we can list all such sentences of the form $\exists x \theta$, enumerating them by the positive integers:

$$\exists x \theta_1, \exists x \theta_2, \exists x \theta_3, \ldots, \exists x \theta_n, \ldots.$$

We will now use the Henkin constants of $\mathcal{L}_1$ to add to Σ countably many axioms, called **Henkin axioms**. These axioms will ensure that every existential sentence that is asserted by Σ will have a witness in our constructed structure $\mathfrak{A}$. The collection of Henkin axioms is

$$H_1 = \{[\exists x \theta_i] \to \theta_i(c_i) \mid (\exists x \theta_i) \text{ is an } \mathcal{L}_0 \text{ sentence}\},$$

where $\theta_i(c_i)$ is shorthand for $\theta^x_{c_i}$.

Now let $\Sigma_0 = \Sigma$, and define

$$\Sigma_1 = \Sigma_0 \cup H_1.$$

> *Chaff:* Foreshadowing!

As Σ_1 contains many more sentences than Σ_0, it seems entirely possible that Σ_1 is no longer consistent. Fortunately, the next lemma shows that is not the case. The proof of the lemma is on page 85.

Lemma 3.2.4. *If Σ_0 is a consistent set of sentences and Σ_1 is created by adding Henkin axioms to Σ_0, then Σ_1 is consistent.*

Now we have Σ_1, a consistent set of $\mathcal{L}_1$-sentences. We can repeat this construction, building a larger language $\mathcal{L}_2$ consisting of $\mathcal{L}_1$ together with an infinite set of new Henkin constants k_i. Then we can let H_2 be a new set of Henkin axioms:

$$H_2 = \{[\exists x \theta_i] \to \theta_i(k_i) \mid (\exists x \theta_i) \text{ is an } \mathcal{L}_1 \text{ sentence}\},$$

and let Σ_2 be $\Sigma_1 \cup H_2$. As before, Σ_2 will be consistent. We can continue this process to build:

- $\mathcal{L} = \mathcal{L}_0 \subseteq \mathcal{L}_1 \subseteq \mathcal{L}_2 \cdots$, an increasing chain of languages.
- $H_1, H_2, H_3, \ldots$, each H_i a collection of Henkin axioms in the language $\mathcal{L}_i$.
- $\Sigma = \Sigma_0 \subseteq \Sigma_1 \subseteq \Sigma_2 \subseteq \cdots$, where each Σ_i is a consistent set of $\mathcal{L}_i$-sentences.

Let $\mathcal{L}' = \bigcup_{i<\infty} \mathcal{L}_i$ and let $\hat{\Sigma} = \bigcup_{i<\infty} \Sigma_i$. Each Σ_i is a consistent set of $\mathcal{L}'$-sentences, as can be shown by proofs that are identical to those of Lemmas 3.2.3 and 3.2.4. You will show in Exercise 2 that $\hat{\Sigma}$ is a consistent set of $\mathcal{L}'$-sentences.

Extending to a Maximal Consistent Set of Sentences

As you recall, we were going to construct our model $\mathfrak{A}$ in such a way that the sentences that were true in $\mathfrak{A}$ were exactly the elements of a set of sentences Σ'. It is time to build Σ'. Since every sentence is either true or false in a given model, it will be necessary for us to make sure that for every sentence $\sigma \in \mathcal{L}'$, either $\sigma \in \Sigma'$ or $\neg\sigma \in \Sigma'$. Since we can't have both σ and $\neg\sigma$ true in any structure, we must also make sure that we don't put both σ and $\neg\sigma$ into Σ'. Thus, Σ' will be a maximal consistent extension of $\hat{\Sigma}$.

To build this extension, fix an enumeration of all of the $\mathcal{L}'$-sentences

$$\sigma_1, \sigma_2, \ldots, \sigma_n, \ldots.$$

We can do this as $\mathcal{L}'$ is countable, being a countable union of countable sets. Now we work our way through this list, one sentence at a time, adding either σ_n or the denial of σ_n to our growing list of sentences, depending on which one keeps our collection consistent.

Here are the details: Let $\Sigma^0 = \hat{\Sigma}$, and assume that Σ^k is known to be a consistent set of $\mathcal{L}'$-sentences. We will show how to build $\Sigma^{k+1} \supseteq \Sigma^k$ and prove that Σ^{k+1} is also a consistent set of $\mathcal{L}'$-sentences. Then we let

$$\Sigma' = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \cdots \cup \Sigma^n \cup \cdots.$$

You will prove in Exercise 4 that Σ' is a consistent set of sentences. It will be obvious from the construction of Σ^{k+1} from Σ^k that Σ' is maximal,

and thus we will have completed our task of producing a maximal consistent extension of $\hat{\Sigma}$.

So all we have to do is describe how to get Σ^{k+1} from Σ^k and prove that Σ^{k+1} is consistent. Given Σ^k, consider the set $\Sigma^k \cup \{\sigma^{k+1}\}$, where σ_{k+1} is the $(k+1)$st element of our fixed list of all of the $\mathcal{L}'$-sentences. Let

$$\Sigma^{k+1} = \begin{cases} \Sigma^k \cup \{\sigma_{k+1}\} & \text{if } \Sigma^k \cup \{\sigma_{k+1}\} \text{ is consistent,} \\ \Sigma^k \cup \{\neg\sigma_{k+1}\} & \text{otherwise.} \end{cases}$$

You are asked in Exercise 3 to prove that Σ^{k+1} is consistent. Once you have done that, we have constructed a maximal consistent Σ' that extends Σ.

The next lemma states that Σ' is deductively closed, at least as far as sentences are concerned. As you work through the proof, the Deduction Theorem will be useful.

Lemma 3.2.5. *If σ is a sentence, then $\sigma \in \Sigma'$ if and only if $\Sigma' \vdash \sigma$.*

Proof. Exercise 5. □

Construction of the Model—Preliminaries

We have mentioned a few times that the model of Σ that we are going to construct will have as its universe the collection of variable-free terms of the language $\mathcal{L}'$. It is now time to confess that we have lied. It is easy to see why the plan of using the terms as the elements of the universe is doomed to failure. Suppose that there are two different terms t_1 and t_2 of the language and somewhere in Σ' is the sentence $t_1 = t_2$. If the *terms* were the elements of the universe, then we could not model Σ', as the two terms t_1 and t_2 are not the same (they are typographically distinct), while Σ' demands that they be equal. Our solution to this problem is to take the collection of variable-free terms, define an equivalence relation on that set, and then construct a model from the *equivalence classes* of the variable-free terms.

So let T be the set of variable-free terms of the language $\mathcal{L}'$, and define a relation $\sim$ on T by

$$t_1 \sim t_2 \text{ if and only if } (t_1 = t_2) \in \Sigma'.$$

It is not difficult to show that $\sim$ is an equivalence relation. We will verify that $\sim$ is symmetric, leaving reflexivity and transitivity to the Exercises.

To show that $\sim$ is symmetric, assume that $t_1 \sim t_2$. We must prove that $t_2 \sim t_1$. As we know $t_1 \sim t_2$, by definition we know that the sentence $(t_1 = t_2)$ is an element of Σ'. We need to show that $(t_2 = t_1) \in \Sigma'$. Assume not. Then by the maximality of Σ', $\neg(t_2 = t_1) \in \Sigma'$. But since we know that $\Sigma' \vdash t_1 = t_2$, by Theorem 2.7.1, $\Sigma' \vdash t_2 = t_1$. (Can you provide the details?) But since we also know that $\Sigma' \vdash \neg(t_2 = t_1)$, it must be the case

that $\Sigma' \vdash \bot$, which is a contradiction, as we know that Σ' is consistent. So our assumption is wrong and $(t_2 = t_1) \in \Sigma'$, and thus $\sim$ is a symmetric relation.

So, assuming that you have worked through Exercise 7, we have established that $\sim$ is an equivalence relation. Now let $[t]$ be the set of all variable-free terms s of the language $\mathcal{L}'$ such that $t \sim s$. So $[t]$ is the equivalence class of all terms that Σ' tells us are equal to t. The collection of all such equivalence classes will be the universe of our model $\mathfrak{A}$.

Construction of the Model—The Main Ideas

To define our model of Σ', we must construct an $\mathcal{L}'$-structure. Thus, we have to describe the universe of our structure as well as interpretations of all of the constant, function, and relation symbols of the language $\mathcal{L}'$. We discuss each of them separately.

The Universe A: As explained above, the universe of $\mathfrak{A}$ will be the collection of $\sim$-equivalence classes of the variable-free terms of $\mathcal{L}'$. For example, if $\mathcal{L}'$ includes the binary function symbol f, the non-Henkin constant symbol k, and the Henkin constants $c_1, c_2, \ldots, c_n, \ldots$, then the universe of our structure would include among its elements $[c_{17}]$ and $[f(k, c_3)]$.

The Constants: For each constant symbol c of $\mathcal{L}'$ (including the Henkin constants), we need to pick out an element $c^{\mathfrak{A}}$ of the universe to be the element represented by that symbol. We don't do anything fancy here:

$$c^{\mathfrak{A}} = [c].$$

So each constant symbol will denote its own equivalence class.

The Functions: If f is an n-ary function symbol, we must define an n-ary function $f^{\mathfrak{A}} : A^n \to A$. Let us write down the definition of $f^{\mathfrak{A}}$ and then we can try to figure out exactly what the definition is saying:

$$f^{\mathfrak{A}}([t_1], [t_2], \ldots, [t_n]) = [f t_1 t_2 \ldots t_n].$$

On the left-hand side of the equality you will notice that there are n equivalence classes that are the inputs to the function $f^{\mathfrak{A}}$. Since the elements of A are equivalence classes and $f^{\mathfrak{A}}$ is an n-ary function, that should be all right. On the right side of the equation there is a single equivalence class, and the thing inside the brackets is a variable-free term of $\mathcal{L}'$. Notice that the function $f^{\mathfrak{A}}$ acts by placing the symbol f in front of the terms and then taking the equivalence class of the result.

There is one detail that has to be addressed. We must show that the function $f^{\mathfrak{A}}$ is well defined. Let us say a bit about what that means,

assuming that f is a unary function symbol, for simplicity. Notice that our definition of $f^{\mathfrak{A}}([t])$ depends on the *name* of the equivalence class that we are putting into $f^{\mathfrak{A}}$. This might lead to problems, as it is at least conceivable that we could have two terms, t_1 and t_2, such that $[t_1]$ is the same set as $[t_2]$, but $f^{\mathfrak{A}}([t_1])$ and $f^{\mathfrak{A}}([t_2])$ evaluate to be different sets. Then our alleged function $f^{\mathfrak{A}}$ wouldn't even be a function. Showing that this does not happen is what we mean when we say that we must show that the function $f^{\mathfrak{A}}$ is well defined.

Let us look at the proof that our function $f^{\mathfrak{A}}$ is, in fact, well defined. Suppose that $[t_1] = [t_2]$. We must show that $f^{\mathfrak{A}}([t_1]) = f^{\mathfrak{A}}([t_2])$. In other words, we must show that if $[t_1] = [t_2]$, then $[ft_1] = [ft_2]$. Again looking at the definition of our equivalence relation $\sim$, this means that we must show that if $t_1 = t_2$ is an element of Σ', then so is $f(t_1) = f(t_2)$. So assume that $t_1 = t_2$ is an element of Σ'. Here is an outline of a deduction from Σ' of $f(t_1) = f(t_2)$:

$x = y \to f(x) = f(y)$	axiom (E2)
$\vdots$	
$t_1 = t_2 \to f(t_1) = f(t_2)$	
$t_1 = t_2$	element of Σ'
$f(t_1) = f(t_2)$	PC

Since $\Sigma' \vdash f(t_1) = f(t_2)$, Lemma 3.2.5 tells us that $f(t_1) = f(t_2)$ is an element of Σ', as needed. So the function $f^{\mathfrak{A}}$ is well defined.

The Relations: Suppose that R is an n-ary relation symbol of $\mathcal{L}'$. We must define an n-ary relation $R^{\mathfrak{A}}$ on A. In other words, we must decide which n-tuples of equivalence classes will stand in the relation $R^{\mathfrak{A}}$. Here is where we use the elements of Σ'. We define $R^{\mathfrak{A}}$ by this statement:

$$R^{\mathfrak{A}}([t_1], [t_2], \ldots, [t_n]) \text{ is true if and only if } Rt_1t_2 \ldots t_n \in \Sigma'.$$

So elements of the universe are in the relation R if and only if Σ' *says* they are in the relation R. Of course, we must show that the relation $R^{\mathfrak{A}}$ is well defined, also. Or rather, **you** must show that the relation $R^{\mathfrak{A}}$ is well defined. See Exercise 8.

At this point we have constructed a perfectly good $\mathcal{L}'$-structure. What we have to do next is show that $\mathfrak{A}$ makes all of the sentences of Σ' true. Then we will have shown that we have constructed a model of Σ'.

Proposition 3.2.6. $\mathfrak{A} \models \Sigma'$.

Proof. We will in fact prove something slightly stronger. We will prove, for each sentence σ, that

$$\sigma \in \Sigma' \text{ if and only if } \mathfrak{A} \models \sigma.$$

Well, since you have noticed, this isn't *really* stronger, as we know that Σ' is maximal. But it does appear stronger, and this version of the proposition is what we need to get the inductive steps to work out nicely.

We proceed by induction on the complexity of the formulas in Σ'. For the base case, suppose that σ is an atomic sentence. Then σ is of the form $Rt_1t_2\ldots t_n$, where R is an n-ary relation symbol and the t_i's are variable free terms. But then our definition of $R^{\mathfrak{A}}$ guaranteed that $\mathfrak{A} \models \sigma$ if and only if $\sigma \in \Sigma'$. Notice that if R is $=$ and σ is $t_1 = t_2$, then $\sigma \in \Sigma'$ iff $t_1 \sim t_2$ iff $[t_1] = [t_2]$ iff $\mathfrak{A} \models \sigma$.

For the inductive cases, suppose first that $\sigma :\equiv \neg\alpha$, where we know by inductive hypothesis that $\mathfrak{A} \models \alpha$ if and only if $\alpha \in \Sigma'$. Notice that as Σ' is a maximal consistent set of sentences, we know that $\sigma \in \Sigma'$ if and only if $\alpha \notin \Sigma'$. Thus

$$\begin{aligned}\sigma \in \Sigma' &\text{ if and only if } \alpha \notin \Sigma' \\ &\text{if and only if } \mathfrak{A} \not\models \alpha \\ &\text{if and only if } \mathfrak{A} \models \neg\alpha \\ &\text{if and only if } \mathfrak{A} \models \sigma.\end{aligned}$$

The second inductive case, when $\sigma :\equiv \alpha \vee \beta$, is similar and is left to the Exercises.

The interesting case is when σ is a sentence of the form $\forall x\phi$. We must show that $\forall x\phi \in \Sigma'$ if and only if $\mathfrak{A} \models \forall x\phi$. We do each implication separately.

First, assume that $\forall x\phi \in \Sigma'$. We must show that $\mathfrak{A} \models \forall x\phi$, which means that we must show, given an assignment function s, that $\mathfrak{A} \models \forall x\phi[s]$. Since the elements of A are equivalence classes of variable-free terms, this means that we have to show for any variable-free term t that

$$\mathfrak{A} \models \phi\big[s[x|[t]]\big].$$

But (here is another lemma for you to prove) for any variable-free term t and any assignment function s, $\overline{s}(t) = [t]$, and so by Theorem 2.6.2, we need to prove that

$$\mathfrak{A} \models \phi^x_t[s].$$

Notice that ϕ^x_t is a sentence, so $\mathfrak{A} \models \phi^x_t[s]$ if and only if $\mathfrak{A} \models \phi^x_t$. But also notice that $\Sigma' \vdash \phi^x_t$, as $\forall x\phi$ is an element of Σ', $\forall x\phi \to \phi^x_t$ is a quantifier axiom of type (Q1) (t is substitutable for x in ϕ as t is variable-free), and Σ' is deductively closed for sentences by Lemma 3.2.5. But ϕ^x_t is less complex than $\forall x\phi$, and thus by our inductive hypothesis, $\mathfrak{A} \models \phi^x_t$, as needed.

For the reverse direction of our biconditional, assume that $\forall x\phi \notin \Sigma'$. We need to show that $\mathfrak{A} \not\models \forall x\phi$. As Σ' is maximal, $\neg\forall x\phi \in \Sigma'$. By deductive closure again, this means that $\exists x\neg\phi \in \Sigma'$. From our construction of Σ', we know there is some Henkin constant c_i such that $\big([\exists x\neg\phi] \rightarrow \neg\phi(c_i)\big) \in \Sigma'$, and using deductive closure once again, this tells us that $\neg\phi(c_i) \in \Sigma'$. Having stripped off a quantifier, we can assert via the inductive hypothesis that $\mathfrak{A} \models \neg\phi(c_i)$, so $\mathfrak{A} \not\models \forall x\phi$, as needed.

This finishes our proof of Lemma 3.2.6, so we know that the $\mathcal{L}'$-structure $\mathfrak{A}$ is a model of Σ'. □

Construction of the Model—Cleaning Up

As you recall, back in our outline of the proof of the Completeness Theorem on page 75, we were going to prove the theorem by constructing a model of Σ. We are almost there. We have a structure, $\mathfrak{A}$, we know that $\mathfrak{A}$ is a model of Σ', and we know that $\Sigma \subseteq \Sigma'$, so every sentence in Σ is true in the structure $\mathfrak{A}$. We're just about done. The only problem is that Σ began life as a set of $\mathcal{L}$-sentences, while $\mathfrak{A}$ is an $\mathcal{L}'$-structure, not an $\mathcal{L}$-structure.

Fortunately, this is easily remedied by a slight bit of amnesia: Define the structure $\mathfrak{A}\restriction_{\mathcal{L}}$ (read $\mathfrak{A}$ **restricted to** $\mathcal{L}$, or the **restriction of** $\mathfrak{A}$ **to** $\mathcal{L}$ as follows: The universe of $\mathfrak{A}\restriction_{\mathcal{L}}$ is the same as the universe of $\mathfrak{A}$. Any constant symbols, function symbols, and relations symbols of $\mathcal{L}$ are interpreted in $\mathfrak{A}\restriction_{\mathcal{L}}$ exactly as they were interpreted in $\mathfrak{A}$, and we just ignore all of the symbols that were added as we moved from $\mathcal{L}$ to $\mathcal{L}'$. Now, $\mathfrak{A}\restriction_{\mathcal{L}}$ is a perfectly good $\mathcal{L}$-structure, and all that is left to finish the proof of the Completeness Theorem is to work through one last lemma:

Lemma 3.2.7. *If σ is an $\mathcal{L}$-sentence, then $\mathfrak{A} \models \sigma$ if and only if $\mathfrak{A}\restriction_{\mathcal{L}} \models \sigma$.*

Proof. (Outline) Use induction on the complexity of σ, proving that $\mathfrak{A}\restriction_{\mathcal{L}} \models \sigma$ if and only if $\sigma \in \Sigma'$, as in the proof of Lemma 3.2.6. □

Thus, we have succeeded in producing an $\mathcal{L}$-structure that is a model of Σ, so we know that every consistent set of sentences has a model. By our preliminary remarks on page 76, we thus know that if $\Sigma \models \phi$, then $\Sigma \vdash \phi$, and our proof of the Completeness Theorem is complete. □

Proofs of the Lemmas

We present here the proofs of two lemmas that were used in the proof of the Completeness Theorem. The first lemma was introduced when we expanded the language $\mathcal{L}$ to the language $\mathcal{L}'$ and we were concerned about the consistency of Σ in the new, expanded language.

(Lemma 3.2.3). *If Σ is a consistent set of $\mathcal{L}$-sentences and $\mathcal{L}'$ is an extension by constants of $\mathcal{L}$, then Σ is consistent when viewed as a set of $\mathcal{L}'$-sentences.*

Proof. Suppose, by way of contradiction, that Σ is not consistent as a set of $\mathcal{L}'$-sentences. Thus there is a deduction (in $\mathcal{L}'$) of $\bot$ from Σ. Let n be the smallest number of new constants used in any such deduction, and let D' be a deduction using exactly n such constants. Notice that $n > 0$, as otherwise D' would be a deduction of $\bot$ in $\mathcal{L}$. We show that there is a deduction of $\bot$ using fewer than n constants, a contradiction that establishes the lemma.

Let v be a variable that does not occur in D', let c be one of the new constants that occurs in D', and let D be the sequence of formulas (ϕ_i) that is formed by taking each formula ϕ'_i in D' and replacing all occurrences of c in ϕ'_i by v. The last formula in D is $\bot$, so if we can show that D is a deduction, we will be finished.

So we use induction on the elements of the deduction D'. If ϕ'_i is an element of D' by virtue of being an equality axiom or an element of Σ, then $\phi_i = \phi'_i$, and ϕ_i is an element of a deduction by the same reason. If ϕ'_i is a quantifier axiom, for example $(\forall x)\theta' \to \theta'^{x}_{t'}$, then ϕ_i will also be a quantifier axiom, in this case $(\forall x)\theta \to \theta^x_t$. There will be no problems with substitutability of t for x, given that t' is substitutable for x. If ϕ' is an element of the deduction by virtue of being the conclusion of a rule of inference (Γ', ϕ'), then (Γ, ϕ) will be a rule of inference that will justify ϕ.

This completes the argument that D is a deduction of $\bot$. Since D clearly uses fewer new constant symbols than D', we have our contradiction and our proof is complete. □

The second lemma was needed when we added the Henkin axioms to our consistent set of sentences Σ. We needed to prove that the resulting set, $\hat{\Sigma}$, was still consistent.

(Lemma 3.2.4). *If Σ is a consistent set of sentences and $\hat{\Sigma}$ is created by adding Henkin axioms to Σ, then $\hat{\Sigma}$ is consistent.*

Proof. Suppose that $\hat{\Sigma}$ is not consistent. Let n be the smallest number of Henkin axioms used in any possible deduction from $\hat{\Sigma}$ of $\bot$. Fix such a set of n Henkin axioms, and let α be one of those Henkin axioms. So we know that

$$\Sigma \cup H \cup \alpha \vdash \bot,$$

where H is the collection of the other $n-1$ Henkin axioms needed in the proof. Now α is of the form $\exists x\phi \to \phi(c)$, where c is a Henkin constant and $\phi(c)$ is our shorthand for ϕ^x_c.

By the Deduction Theorem (Theorem 2.7.4), as α is a sentence, this means that $\Sigma \cup H \vdash \neg\alpha$, so

$$\Sigma \cup H \vdash \exists x\phi \quad \text{and} \quad \Sigma \cup H \vdash \neg\phi^x_c.$$

Since $\exists x\phi$ is the same as $\neg\forall x\neg\phi$, from the first of these facts we know that

$$\Sigma \cup H \vdash \neg\forall x\neg\phi. \tag{3.1}$$

We also know that $\Sigma \cup H \vdash \neg\phi_c^x$. If we take a deduction of $\neg\phi_c^x$ and replace each occurrence of the constant c by a new variable z, the result is still a deduction (as in the proof of Lemma 3.2.3 above), so $\Sigma \cup H \vdash \neg\phi_z^x$. By Lemma 2.7.2, we know that

$$\Sigma \cup H \vdash \forall z \neg\phi_z^x.$$

Our quantifier axiom (Q1) states that as long as x is substitutable for z in $\neg\phi_z^x$, (which it is, as z is a *new* variable), then we may assert that $\left[\forall z \neg\phi_z^x\right] \to \neg(\phi_z^x)_x^z$. Therefore

$$\Sigma \cup H \vdash \neg(\phi_z^x)_x^z.$$

But $(\phi_z^x)_x^z = \phi$, so $\Sigma \cup H \vdash \neg\phi$. But now we can use Lemma 2.7.2 again to conclude that

$$\Sigma \cup H \vdash \forall x \neg\phi. \tag{3.2}$$

So, by Equations (3.1) and (3.2), we see that $\Sigma \cup H \vdash \bot$. This is a contradiction, as $\Sigma \cup H$ contains only $n-1$ Henkin axioms. Thus we are led to conclude that $\hat{\Sigma}$ is consistent. □

3.2.1 Exercises

1. Suppose that Σ is inconsistent and ϕ is an $\mathcal{L}$-formula. Prove that $\Sigma \vdash \phi$.

2. Assume that $\Sigma_0 \subseteq \Sigma_1 \subseteq \Sigma_2 \cdots$ are such that each Σ_i is a consistent set of sentences in a language $\mathcal{L}$. Show $\bigcup \Sigma_i$ is consistent.

3. Show that if Π is any consistent set of sentences and σ is a sentence such that $\Pi \cup \{\sigma\}$ is inconsistent, then $\Pi \cup \{\neg\sigma\}$ is consistent. Conclude that in the proof of the Completeness Theorem, if Σ^k is consistent, then Σ^{k+1} is consistent.

4. Prove that the Σ' constructed in the proof of the Completeness Theorem is consistent. [*Suggestion:* Deductions are finite in length.]

5. Prove Lemma 3.2.5.

6. Toward a proof of the Completeness Theorem in a more general setting:

   (a) Do not assume that the language $\mathcal{L}$ is countable. Suppose that you have been given a set of sentences $\Sigma_{\max}$ that is maximal and consistent. So for each sentence σ, either $\sigma \in \Sigma_{\max}$ or $\neg\sigma \in \Sigma_{\max}$. Mimic the proof of Proposition 3.2.6 to convince yourself that we can construct a model $\mathfrak{A}$ of Σ.

   (b) Zorn's Lemma implies the following: If we are given a consistent set of $\mathcal{L}'$-sentences $\hat{\Sigma}$, then the collection of consistent extensions of $\hat{\Sigma}$ has a maximal (with respect to $\subseteq$) element $\Sigma_{\max}$. If you are familiar with Zorn's Lemma, prove this fact.

(c) Use parts (a) and (b) of this problem to outline a proof of the Completeness Theorem in the case where the language $\mathcal{L}$ is not countable.

7. Complete the proof of the claim on page 80 that the relation $\sim$ is an equivalence relation.

8. Show that the relation $R^{\mathfrak{A}}$ of the structure $\mathfrak{A}$ is well defined. So let R be a relation symbol (a unary relation symbol is fine), and show that if $[t_1] = [t_2]$, then $R^{\mathfrak{A}}([t_1])$ is true if and only if $R^{\mathfrak{A}}([t_2])$ is true.

9. Finish the inductive clause of the proof of Proposition 3.2.6.

10. Fill in the details of the proof of Lemma 3.2.7.

3.3 Compactness

The Completeness Theorem finishes our link between deducibility and logical implication. The Compactness Theorem is our first use of that link. In some sense, what the Compactness Theorem does is focus our attention on the finiteness of deductions, and then we can begin to use that finiteness to our advantage.

Theorem 3.3.1 (Compactness Theorem). *Let Σ be any set of axioms. There is a model of Σ if and only if every finite subset Σ_0 of Σ has a model.*

We say that Σ is **satisfiable** if there is a model of Σ, and we say that Σ is **finitely satisfiable** if every finite subset of Σ has a model. So the Compactness Theorem says that Σ is satisfiable if and only if Σ is finitely satisfiable.

Proof. For the easy direction, suppose that Σ has a model $\mathfrak{A}$. Then $\mathfrak{A}$ is also a model of every finite $\Sigma_0 \subseteq \Sigma$.

For the more difficult direction, assume there is no model of Σ. Then $\Sigma \models \bot$. By the Completeness Theorem, $\Sigma \vdash \bot$, so there is a deduction D of $\bot$ from Σ. Since D is a deduction, it is finite in length and thus can only contain finitely many of the axioms of Σ. Let Σ_0 be the finite set of axioms from Σ that are used in D. Then D is a deduction from Σ_0, so $\Sigma_0 \vdash \bot$. But then by the Soundness Theorem, $\Sigma_0 \models \bot$, so Σ_0 cannot have a model. □

Corollary 3.3.2. *Let Σ be a set of $\mathcal{L}$-formulas and let θ be an $\mathcal{L}$-formula. $\Sigma \models \theta$ if and only if there is a finite $\Sigma_0 \subseteq \Sigma$ such that $\Sigma_0 \models \theta$.*

Proof.

$\Sigma \models \theta$ iff $\Sigma \vdash \theta$	Soundness and Completeness
iff $\Sigma_0 \vdash \theta$ for a finite $\Sigma_0 \subseteq \Sigma$	deductions are finite
iff $\Sigma_0 \models \theta$	Soundness and Completeness

□

Now we are in a position where we can use the Compactness Theorem to get a better understanding of the limitations of first-order logic—or, to put a more positive spin on it, a better understanding of the richness of mathematics!

Example 3.3.3. Suppose that we examine the $\mathcal{L}_{NT}$-structure $\mathfrak{N}$, whose universe is the set of natural numbers $\mathbb{N}$, endowed with the familiar arithmetic functions of addition, multiplication, and exponentiation and the usual binary relation less than. It would be nice to have a collection of axioms that would characterize the structure $\mathfrak{N}$. By this we mean a set of sentences Σ such that $\mathfrak{N} \models \Sigma$, and if $\mathfrak{A}$ is any $\mathcal{L}_{NT}$-structure such that $\mathfrak{A} \models \Sigma$, then $\mathfrak{A}$ is "just like" $\mathfrak{N}$. ($\mathfrak{A}$ is "just like" $\mathfrak{N}$ if there $\mathfrak{A}$ and $\mathfrak{N}$ are isomorphic—see Exercise 5 in Section 1.6.1).

Unfortunately, we cannot hope to have such a set of sentences, and the Compactness Theorem shows us why. Suppose we took any set of sentences Σ that seemed like it ought to characterize $\mathfrak{N}$. Let us add some sentences to Σ and create a new collection of sentences Θ in an extended language $\mathcal{L} = \mathcal{L}_{NT} \cup \{c\}$, where c is a new constant symbol:

$$\Theta = \Sigma \cup \{0 < c, S0 < c, SS0 < c, \ldots, \underbrace{SSS\cdots S}_{n S\text{'s}}0 < c, \ldots\}.$$

Now notice that Θ is finitely satisfiable: If Θ_0 is a finite subset of Θ, then Θ_0 is a subset of

$$\Theta_n = \Sigma \cup \{0 < c, S0 < c, SS0 < c, \ldots, \underbrace{SSS\cdots S}_{n S\text{'s}}0 < c\}$$

for some natural number n. But Θ_n has a model $\mathfrak{N}_n$, whose universe is $\mathbb{N}$, the functions and relations are interpreted in the usual way, and $c^{\mathfrak{N}_n} = n+1$. So every finite subset of Θ has a model, and thus Θ has a model $\mathfrak{A}'$. Now forget the interpretation of the constant symbol c and you are left with an $\mathcal{L}_{NT}$-structure $\mathfrak{A} = \mathfrak{A}' \restriction_{\mathcal{L}_{NT}}$. This model $\mathfrak{A}$ is interesting, but we cannot claim that $\mathfrak{A}$ is "just like" $\mathfrak{N}$, since $\mathfrak{A}$ has an element (the thing that used to be called $c^{\mathfrak{A}'}$) such that there are infinitely many elements x that stand in the relation $<$ with that element, while there is no such element of $\mathfrak{N}$. The element $c^{\mathfrak{A}'}$ is called a nonstandard element of the universe, and $\mathfrak{A}$ is another example of a nonstandard model of arithmetic, a model of arithmetic that is not isomorphic to $\mathfrak{N}$. We first encountered nonstandard models of arithmetic in Exercise 7 of Section 2.8.1.

So no set of first-order sentences can completely characterize the natural numbers.

> *Chaff:* Isn't this neat! Notice how each of the $\mathfrak{N}_n$'s in the last example were perfectly ordinary models that looked just like the natural numbers, but the thing that we got at the end looked entirely different!

Definition 3.3.4. If $\mathfrak{A}$ is an $\mathcal{L}$-structure, we define the **theory of** $\mathfrak{A}$ to be $Th(\mathfrak{A}) = \{\phi \mid \phi \text{ is an } \mathcal{L}\text{-formula and } \mathfrak{A} \models \phi\}$. If $\mathfrak{A}$ and $\mathfrak{B}$ are $\mathcal{L}$-structures such that $Th(\mathfrak{A}) = Th(\mathfrak{B})$, then we say that $\mathfrak{A}$ and $\mathfrak{B}$ are **elementarily equivalent**, and write $\mathfrak{A} \equiv \mathfrak{B}$.

If $\mathfrak{A} \equiv \mathfrak{N}$, we say that $\mathfrak{A}$ is a **model of arithmetic**

Example (continued). Notice that the weird structure $\mathfrak{A}$ that we constructed above can be a model of arithmetic if we just let the Σ of our construction be $Th(\mathfrak{N})$. Exercise 2 asks you to prove that in this case we have $\mathfrak{A} \equiv \mathfrak{N}$. Since $\mathfrak{A}$ certainly is not anything like the usual model of arithmetic on the natural numbers, calling $\mathfrak{A}$ a nonstandard model of arithmetic makes pretty good sense. The difficult thing to see is that although the universe A certainly contains nonstandard elements, they don't get in the way of elementary equivalence. The reason for this is that the language $\mathcal{L}_{NT}$ can't refer to any nonstandard element explicitly, so we can't express a statement that is (for example) true in $\mathfrak{N}$ but false in $\mathfrak{A}$. So the lesson to be learned is that it is much easier for two structures to be elementarily equivalent than it is for them to be isomorphic: Our structure $\mathfrak{A}$ is not isomorphic to $\mathfrak{N}$, but $\mathfrak{A}$ is elementarily equivalent to $\mathfrak{N}$.

Example 3.3.5. Remember those ϵ's and δ's from calculus? They were introduced in the nineteenth century in an attempt to firm up the foundations of the subject. When they were developing the calculus, Newton and Leibniz did not worry about limits. They happily used quantities that were infinitely small but not quite zero and they ignored the logical difficulties this presented. These infinitely small quantities live on in today's calculus textbooks as the differentials dx and dy.

Most people find thinking about differentials much easier than fighting through limit computations, and in 1961 Abraham Robinson developed a logical framework for calculus that allowed the use of these infinitesimals in a coherent, noncontradictory way. Robinson's version of the calculus came to be known as nonstandard analysis. Here is a rough introduction (for a complete treatment, see [Keisler 76]).

Taking as our starting point the real numbers that you know so well, we construct a language $\mathcal{L}_{\mathbb{R}}$, the language of the real numbers. For each real number r, the language $\mathcal{L}_{\mathbb{R}}$ includes a constant symbol $\dot{r}$. So the language $\mathcal{L}_{\mathbb{R}}$ includes constant symbols $\dot{0}, \dot{\pi}$, and $\dot{\frac{2}{7}}$. For each function $f : \mathbb{R}^n \to \mathbb{R}$, we toss in a function symbol $\dot{f}$, and for each n-ary relation R on the reals we add an n-ary relation symbol $\dot{R}$. So our language includes, for example, the function symbols $\dot{+}$ and $\dot{\cos}$ and the relation symbol $\dot{<}$.

Now we define $\mathfrak{R}$ to be the $\mathcal{L}_{\mathbb{R}}$-structure $(\mathbb{R}, \{r\}, \{f\}, \{R\})$, where each symbol is interpreted as meaning the number, function, or relation that gave rise to the symbol. So the function symbol $\dot{+}$ stands for the function addition, and the constant symbol $\dot{\pi}$ refers to the real number that is equal to the ratio of the circumference of a circle to its diameter.

Given this structure $\mathfrak{R}$ (notice that $\mathfrak{R}$ is not anything fancy—it is just the real numbers you have been working with since high school), it generates the set of formulas $Th(\mathfrak{R})$, the collection of first-order $\mathcal{L}_{\mathbb{R}}$-formulas that are true statements about the real numbers. Now it is time to use compactness.

Let $\mathcal{L}' = \mathcal{L}_{\mathbb{R}} \cup \{c\}$, where c is a new constant symbol, and look at the collection of $\mathcal{L}'$-sentences

$$\Theta = Th(\mathfrak{R}) \cup \{\dot{0} \dot{<} c\} \cup \{c \dot{<} \dot{r} \mid r \in \mathbb{R}, r > 0\}.$$

(Are you clear about the difference between the dotted and the undotted symbols in this definition?)

By the Compactness Theorem, Θ has a model, $\mathfrak{A}$, and in the model $\mathfrak{A}$, the element denoted by c plays the role of an infinitesimal element: It is positive, yet it is smaller than every positive real number. Speaking roughly, in the universe A of the structure $\mathfrak{A}$ there are three kinds of elements. There are pure standard elements, which constitute a copy of $\mathbb{R}$ that lives inside A. Then there are pure nonstandard elements, for example, the element denoted by c. Finally, there are elements such as the object denoted by $\dot{17}\dot{+}c$, which has a standard part and a nonstandard part. (For more of the details, see Exercise 11 in Section 3.4.1.)

The nonstandard elements of the structure $\mathfrak{R}$ provide a method for developing derivatives without using limits. For example, we can define the derivative of a function f at a standard element a to be

$$f'(a) = \text{the standard part of } \frac{f(a+c) - f(a)}{c}.$$

As you can see, there is no limit in the definition. We have traded the limits of calculus for the nonstandard elements of $\mathfrak{A}$, and the slope of a tangent line is nothing more than a slope of a line connecting two points, one of which is not standard. Nonstandard analysis has been an area of active study for the past forty years, and although it is not exactly mainstream, it has been used to discover some new results in various areas of classical analysis.

Example 3.3.6. The idea of coloring a map is supposed to be intuitive. When you were in geography class as a child, you were doubtless given a map of a region and asked to color in the various countries, or states, or provinces. And you were missing the point if you used the same color to shade two countries that shared a common border, although it was permitted to use the same color for two countries whose borders met at a single point. (The states of Utah, Colorado, Arizona, and New Mexico do this in the United States, so coloring both Colorado and Arizona with the color red would be permitted.) The question of how many colors are needed to color any map drawn on the plane was first posed in 1852 by Francis Guthrie, and the answer, that four colors suffice for any such map (as long as each political division consists of a single region—Michigan in a map

of the United States or pre-1971 Pakistan in a map of Asia would not be permitted), was proven in 1976 by Kenneth Appel and Wolfgang Hakin. We are not going to prove the Four-Color Theorem here; rather, we extend this result by considering maps with infinitely many regions.

Let R be a set (I'm thinking of the elements of R as being the regions of a map with infinitely many countries) with a symmetric binary relation A (adjacency). Let k be a natural number. We claim that it is possible to assign to each region of R one of k possible colors in such a way that adjacent regions receive different colors if and only if it is possible to so color each finite subset of R.

We will prove this using the Compactness Theorem. One of the tricks to using compactness is to choose your language wisely. For this example, let the language $\mathcal{L}$ consist of a collection of constants $\{r\}_{r\in R}$, one for each region, and a collection of unary predicates $\{C_i\}_{1\leq i\leq k}$, one for each color. So the atomic statement $C_i(r)$ will be interpreted as meaning that region r gets colored with color i. We will also need a binary relation symbol A, for adjacency.

Let Σ be the collection of sentences:

$$\Sigma = \begin{cases} C_1(r) \vee C_2(r) \vee \cdots \vee C_k(r) & \text{for each } r \in R \\ \neg[C_i(r) \wedge C_j(r)] & r \in R, i \neq j \\ A(r,r') \to (\neg C_i(r) \wedge C_i(r')) & r, r' \in R, 1 \leq i \leq k \\ A(r,r') & r, r' \in R, r \text{ adjacent to } r' \\ \neg A(r,r') & r, r' \in R, r \text{ not adjacent to } r'. \end{cases}$$

> *Chaff:* Stop now for a minute and make sure that you understand each of the sentences in Σ. You ought to be able to say, in ordinary English, what each sentence asserts. For example, $C_1(r) \vee C_2(r) \vee \cdots \vee C_k(r)$ says that region r must be given one of the k colors. In other words, we have to color each region on the map. Take the time now to translate each of the other statement types of Σ into English.

But now our claim that an infinite map is k-colorable if and only if each finite subset of the map is k-colorable is clear, as a coloring of (a finite subset of) R corresponds to a model of (a finite subset of) Σ, and the Compactness Theorem says that Σ has a model if and only if every finite subset of Σ has a model.

Notice that no quantifiers are used in this example, so we really only needed compactness for predicate logic, not first-order logic. If you are comfortable with the terms, notice also that the proof works whether there are a countably infinite or an uncountably infinite collection of countries.

If you have really been paying attention, you noticed that we did not use the fact that the maps are drawn on the plane. So if we draw a map

on a donut with uncountably many countries, it only takes seven colors to color the map, as it was proven in 1890 by Percy John Heawood that seven colors suffice for finite maps drawn on a donut.

Example 3.3.7. You may well be familiar with mathematical trees, as they are often discussed in courses in discrete mathematics or introductory computer science courses. For our purposes a **tree** is a set T partitioned into subsets $T_i, (i = 0, 1, 2, \ldots)$, called the levels of the tree, together with a function a such that:

1. T_0 consists of a single element (called the root of the tree).

2. $a : (T - T_0) \to T$ such that if $t \in T_i, i > 0$, then $a(t) \in T_{i-1}$.

A **path** through T consists of a subset $P \subseteq T$ such that $P \cap T_i$ contains exactly one element for each i and P is closed under a. If $t \in T$, the **immediate predecessor** of t is $a(t)$. And an element t_2 is said to be a **predecessor** of t_1 if $t_2 = \underbrace{a(a(\cdots a}_{k\ a\text{'s}}(t_1)))$ for some $k \geq 1$.

We can now use the Compactness Theorem to prove

Lemma 3.3.8 (König's Infinity Lemma). *Let T be a tree all of whose levels are finite and nonempty. Then there is a path through T.*

Proof. Suppose that we are given such a tree T. Let $\mathcal{L}$ be the language consisting of one constant symbol $\hat{t}$ for each element $t \in T$, a unary relation symbol Q, which will be true for elements on the path, and one unary function symbol p, where $p(\hat{t_i})$ is intended to be the immediate predecessor of t_i.

Let Σ be the following set of $\mathcal{L}$-formulas:

$$\Sigma = \begin{cases} p(\hat{t_1}) = \hat{t_2} & \text{for each } t_1, t_2 \in T \text{ such that } a(t_1) = t_2 \\ Q(\hat{t_1}) \vee \cdots \vee Q(\hat{t_k}) & \text{where } T_n = \{t_1, t_2, \ldots, t_k\} \text{ (for each } n) \\ \neg(Q(\hat{t_1}) \wedge Q(\hat{t_2})) & \text{for } t_1, t_2 \in T_n, t_1 \neq t_2 \\ Q(\hat{t}) \to Q(p(\hat{t})) & \text{for each } t \in T - T_0. \end{cases}$$

We claim that Σ is finitely satisfiable: Let Σ_0 be a finite subset of Σ, and let n be so large that if $\hat{t}$ is mentioned in Σ_0, then $t \in T_0 \cup T_1 \cup \cdots \cup T_n$. Pick any element $t^* \in T_{n+1}$, and build an $\mathcal{L}$-structure $\mathfrak{A}$ by letting the universe A be the tree T, $\hat{t}^{\mathfrak{A}}$ be t, letting $p^{\mathfrak{A}}$ be the function a, and letting $Q^{\mathfrak{A}}$ be the collection of predecessors of t^*. It is easy to check that $\mathfrak{A}$ is a model of Σ_0, and thus by compactness, there is a structure $\mathfrak{B}$ such that $\mathfrak{B}$ is a model of Σ. If we let $P = \{t \in T \mid \hat{t}^{\mathfrak{B}} \in Q^{\mathfrak{B}}\}$, then P is a path through T, and König's Infinity Lemma is proven. $\square$

3.3.1 Exercises

1. A common attempt to try to write a set of axioms that would characterize $\mathfrak{N}$ (see Example 3.3.3) is to let Σ be the collection of *all* $\mathcal{L}_{NT}$-formulas that are true in $\mathfrak{N}$, and then to argue that this is an element of Σ:

$$(\forall x)(\exists n)(x = \underbrace{SSS\cdots S}_{n S\text{'s}}0).$$

   Therefore, there can be no nonstandard elements in any model of Σ. Explain why this reasoning fails.

2. Show that if we let $\Sigma = Th(\mathfrak{N})$ in the construction of Example 3.3.3, then the structure $\mathfrak{A}$ that is constructed is elementarily equivalent to the structure $\mathfrak{N}$. Thus $\mathfrak{A}$ is a model of arithmetic.

3. Show that if $\mathfrak{A}$ and $\mathfrak{B}$ are $\mathcal{L}$-structures such that $\mathfrak{A} \cong \mathfrak{B}$, then $\mathfrak{A} \equiv \mathfrak{B}$.

4. Suppose that Σ is a set of $\mathcal{L}$-sentences such that at least one sentence from Σ is true in each $\mathcal{L}$-structure. Show that the disjunction of some finitely many sentences from Σ is logically valid.

5. Show that every nonstandard model of arithmetic contains an infinite prime number, that is, an infinite number a such that if $a = bc$, then either $b = 1$ or $c = 1$.

6. Show that if $\phi(x)$ is a formula with one free variable in $\mathcal{L}_{NT}$ such that there are infinitely many natural numbers a such that $\mathfrak{N} \models \phi(x)[s[x|a]]$, then in every nonstandard model of arithmetic $\mathfrak{N}^*$ there is an infinite number b such that $\mathfrak{N}^* \models \phi(x)[s[x|b]]$.

7. Verify that we can use the Compactness Theorem in Example 3.3.5 by verifying that every finite subset of Θ has a model.

8. (a) Using only connectives, quantifiers, variables, and the equality symbol, construct a set of sentences Σ such that every model of Σ is infinite.

   (b) Prove that if Γ is a set of sentences with arbitrarily large finite models, then Γ has an infinite model.

   (c) Show that there can be no set of sentences in first-order logic that characterizes the finite groups. (See Exercise 3 in Section 2.8.1.)

   (d) Prove that there is no finite set of sentences

$$\Phi = \{\phi_1, \phi_2, \ldots, \phi_n\}$$

   such that $\mathfrak{A} \models \Phi$ if and only if A is infinite. [*Suggestion:* Look at $\neg(\phi_1 \wedge \phi_2 \wedge \cdots \wedge \phi_n)$.]

9. Suppose that Σ_1 and Σ_2 are two sets of sentences such that no structure is a model of both Σ_1 and Σ_2. Show there is a sentence α such that every model of Σ_1 is also a model of α and furthermore, every model of Σ_2 is a model of $\neg\alpha$.

10. A binary relation $<$ on a set A is said to be a **linear order** if

    (a) $<$ is irreflexive—$(\forall a \in A)(\neg a < a)$.

    (b) $<$ is transitive—$(\forall a, b, c \in A)\big([a < b \wedge b < c] \rightarrow a < c\big)$.

    (c) $<$ satisfies trichotomy—$\forall a, b \in A$ exactly one of the following is true: $a < b, b < a,$ or $a = b$.

    If a linear order $<$ has the additional property that there are no infinite descending chains—there do not exist $a_1, a_2, \ldots \in A$ such that $a_1 > a_2 > a_3 > \cdots$ (where $a_1 > a_2$ means $a_2 < a_1$), then the relation $<$ is a **well-order** of the set A. Suppose that $\mathcal{L}$ is a language containing a binary relation symbol $<$. Show there is no set of $\mathcal{L}$-sentences Σ such that Σ has both of the following properties:

    (a) Σ has an infinite model $\mathfrak{A}$ in which $<^{\mathfrak{A}}$ is a linear order of A.

    (b) If $\mathfrak{B}$ is any infinite model of Σ, then $<^{\mathfrak{B}}$ is a well-ordering of B.

11. Show that $<$ is not a well-order in any nonstandard model of arithmetic.

12. (a) In the structure $\mathfrak{A}$ that was built in Example 3.3.5, explain how we know that

$$\mathfrak{A} \models (\forall x)\big[(x \dot{>} \dot{0}) \rightarrow (x \dot{/} \dot{2} \dot{>} \dot{0} \wedge x \dot{>} x \dot{/} \dot{2})\big].$$

    (b) Show that $<$ is a linear order of A, the universe of $\mathfrak{A}$.

    (c) Show that $<$ is not a well-order in this structure.

3.4 Substructures and the Löwenheim–Skolem Theorems

In this section we will discuss a relation between structures. A given set of sentences may have many different models, and it will turn out that in some cases those models are related in surprising ways. We begin by defining the notion of a substructure.

Definition 3.4.1. If $\mathfrak{A}$ and $\mathfrak{B}$ are two $\mathcal{L}$-structures, we will say that $\mathfrak{A}$ is a **substructure** of $\mathfrak{B}$, and write $\mathfrak{A} \subseteq \mathfrak{B}$, if:

1. $A \subseteq B$.
2. For every constant symbol c, $c^{\mathfrak{A}} = c^{\mathfrak{B}}$.

3. For every n-ary relation symbol R, $R^{\mathfrak{A}} = R^{\mathfrak{B}} \cap A^n$.

4. For every n-ary function symbol f, $f^{\mathfrak{A}} = f^{\mathfrak{B}} \restriction_{A^n}$. In other words, for every n-ary function symbol f and every $a \in A$, $f^{\mathfrak{A}}(a) = f^{\mathfrak{B}}(a)$. (This is called the **restriction of the function $f^{\mathfrak{B}}$ to the set A^n**.)

Thus a substructure of $\mathfrak{B}$ is completely determined by its universe, and this universe can be any nonempty subset of B that contains the constants and is closed under every function f.

Example 3.4.2. Suppose that we try to build a substructure $\mathfrak{A}$ of the structure $\mathfrak{N} = (\mathbb{N}, 0, S, +, \cdot, E, <)$. Since A must be closed under the functions and contain the constants, the number 0 must be an element of the universe A. But now, since the substructure must be closed under the function S, it is clear that every natural number must be an element of A. Thus $\mathfrak{N}$ has no proper substructures.

Example 3.4.3. Now, suppose that we try to find some substructures of the structure $\mathfrak{B} = (\mathbb{N}, 0, <)$, with the usual interpretations of 0 and $<$. Since there are no function symbols, any nonempty subset of $\mathbb{N}$ that includes the number 0 can serve as the universe of a substructure $\mathfrak{A} \subseteq \mathfrak{B}$.

Suppose that we let $\mathfrak{A} = (\{0\}, 0, <)$. Then notice that even though $\mathfrak{A} \subseteq \mathfrak{B}$, there are plenty of sentences that are true in one structure that are not true in the other structure. For example, $(\forall x)(\exists y)x < y$ is false in $\mathfrak{A}$ and true in $\mathfrak{B}$. It will not be hard for you to find an example of a sentence that is true in $\mathfrak{A}$ and false in $\mathfrak{B}$.

As Example 3.4.3 shows, if we are given two structures such that $\mathfrak{A} \subseteq \mathfrak{B}$, most of the time you would expect that $\mathfrak{A}$ and $\mathfrak{B}$ would be very different, and there would be lots of sentences that would be true in one of the structures that would not be true in the other.

Sometimes, however, truth in the smaller structure is more closely tied to truth in the larger structure.

Definition 3.4.4. Suppose that $\mathfrak{A}$ and $\mathfrak{B}$ are $\mathcal{L}$-structures and $\mathfrak{A} \subseteq \mathfrak{B}$. We say that $\mathfrak{A}$ is an **elementary substructure** of $\mathfrak{B}$ (equivalently, $\mathfrak{B}$ is an **elementary extension** of $\mathfrak{A}$), and write $\mathfrak{A} \prec \mathfrak{B}$, if for every $s : \mathit{Vars} \to A$ and for every $\mathcal{L}$-formula ϕ,

$$\mathfrak{A} \models \phi[s] \text{ if and only if } \mathfrak{B} \models \phi[s].$$

Chaff: Notice that if we want to prove $\mathfrak{A} \prec \mathfrak{B}$, we need only prove $\mathfrak{A} \models \phi[s] \to \mathfrak{B} \models \phi[s]$, since once we have done that, the other direction comes for free by using the contrapositive and negations.

Proposition 3.4.5. *Suppose that $\mathfrak{A} \prec \mathfrak{B}$. Then a sentence σ is true in $\mathfrak{A}$ if and only if it is true in $\mathfrak{B}$.*

Proof. Exercise 5. □

Example 3.4.6. We saw earlier that the structure $\mathfrak{B} = (\mathbb{N}, 0, <)$ has lots of substructures. However, $\mathfrak{B}$ has no proper elementary substructures. For suppose that $\mathfrak{A} \prec \mathfrak{B}$. Certainly, $0 \in A$, as $\mathfrak{A}$ is a substructure. Since the sentence $(\exists y)\big[0 < y \wedge (\forall x)(0 < x \to y \leq x)\big]$ is true in $\mathfrak{B}$, it must be true in $\mathfrak{A}$ as well. So

$$\mathfrak{A} \models (\exists y)\big[0 < y \wedge (\forall x)(0 < x \to y \leq x)\big].$$

Thus, for any assignment function $s : \mathit{Vars} \to A$ there is some $a \in A$ such that

$$\mathfrak{A} \models \big[0 < y \wedge (\forall x)(0 < x \to y \leq x)\big][s[y|a]].$$

Fix such an s and such an $a \in A$. Now we use elementarity again. Since $\mathfrak{A} \prec \mathfrak{B}$ and $s[y|a] : \mathit{Vars} \to A$, we know that

$$\mathfrak{B} \models \big[0 < y \wedge (\forall x)(0 < x \to y \leq x)\big][s[y|a]].$$

But in the structure $\mathfrak{B}$, there is a unique element that makes the formula $\big[0 < y \wedge (\forall x)(0 < x \to y \leq x)\big]$ true, namely the number 1. So a must be the number 1, and so 1 must be an element of A. Similarly, you can show that $2 \in A$, $3 \in A$, and so on. Thus $\mathbb{N} \subseteq A$, and $\mathfrak{A}$ will not be a proper elementary substructure of $\mathfrak{B}$.

This example shows that when building an elementary substructure of a given structure $\mathfrak{B}$, we need to make sure that witnesses for each existential sentence true in $\mathfrak{B}$ must be included in the universe of the elementary substructure $\mathfrak{A}$. That idea will be the core of the proof of the Downward Löwenheim–Skolem Theorem, Theorem 3.4.8. In fact, the next lemma says that making sure that such witnesses are elements of $\mathfrak{A}$ is all that is needed to ensure that $\mathfrak{A}$ is an elementary substructure of $\mathfrak{B}$.

Lemma 3.4.7. *Suppose that $\mathfrak{A} \subseteq \mathfrak{B}$ and that for every formula α and every $s : \mathit{Vars} \to A$ such that $\mathfrak{B} \models \exists x \alpha[s]$ there is an $a \in A$ such that $\mathfrak{B} \models \alpha[s[x|a]]$. Then $\mathfrak{A} \prec \mathfrak{B}$.*

Proof. We will show, given the assumptions of the lemma, that if ϕ is any formula and s is any variable assignment function into A, $\mathfrak{A} \models \phi[s]$ if and only if $\mathfrak{B} \models \phi[s]$, and thus $\mathfrak{A} \prec \mathfrak{B}$.

This is an easy proof by induction on the complexity of ϕ, which we will make even easier by noting that we can replace the $\forall$ inductive step by an $\exists$ inductive step, as $\forall$ can be defined in terms of $\exists$.

So for the base case, assume that ϕ is atomic. For example, if ϕ is $R(x, y)$, then $\mathfrak{A} \models \phi[s]$ if and only if $(s(x), s(y)) \in R^{\mathfrak{A}}$. But $R^{\mathfrak{A}} = R^{\mathfrak{B}} \cap A^2$,

so $(s(x), s(y)) \in R^{\mathfrak{A}}$ if and only if $(s(x), s(y)) \in R^{\mathfrak{B}}$. But $(s(x), s(y)) \in R^{\mathfrak{B}}$ if and only if $\mathfrak{B} \models \phi[s]$, as needed.

For the inductive clauses, assume that ϕ is $\neg\alpha$. Then

$$\begin{aligned}
\mathfrak{A} \models \phi[s] &\text{ if and only if } \mathfrak{A} \models \neg\alpha[s] \\
&\text{if and only if } \mathfrak{A} \not\models \alpha[s] \\
&\text{if and only if } \mathfrak{B} \not\models \alpha[s] \quad \text{inductive hypothesis} \\
&\text{if and only if } \mathfrak{B} \models \neg\alpha[s] \\
&\text{if and only if } \mathfrak{B} \models \phi[s].
\end{aligned}$$

The second inductive clause, if ϕ is $\alpha \vee \beta$, is similar.

For the last inductive clause, suppose that ϕ is $\exists x\alpha$. Suppose also that $\mathfrak{A} \models \phi[s]$; in other words, $\mathfrak{A} \models \exists x\alpha[s]$. Then, for some $a \in A$, $\mathfrak{A} \models \alpha[s[x|a]]$. Since $s[x|a]$ is a function mapping variables into A, by our inductive hypothesis, $\mathfrak{B} \models \alpha[s[x|a]]$. But then $\mathfrak{B} \models \exists x\alpha[s]$, as needed. For the other direction, assume that $\mathfrak{B} \models \exists x\alpha[s]$, where $s : \mathit{Vars} \to A$. We use the assumption of the lemma to find an $a \in A$ such that $\mathfrak{B} \models \alpha[s[x|a]]$. As $s[x|a]$ is a function with codomain A, by the inductive hypothesis $\mathfrak{A} \models \alpha[s[x|a]]$, and thus $\mathfrak{A} \models \exists x\alpha[s]$, and the proof is complete. □

> *Chaff:* We are now going to look at the Löwenheim–Skolem Theorems, which were published in 1915. To understand these theorems, you need to have at least a basic understanding of cardinality, a topic that is outlined in the Appendix. However, if you are in a hurry, it will suffice if you merely remember that there are many different sizes of infinite sets. An infinite set A is countable if there is a bijection between A and the set of natural numbers $\mathbb{N}$, otherwise, the set is uncountable. Examples of countable sets include the integers and the set of rational numbers. The set of real numbers is uncountable, in that there is no bijection between $\mathbb{R}$ and $\mathbb{N}$. So there are more reals than natural numbers. There are infinitely many different sizes of infinite sets. The smallest infinite size is countable.

Theorem 3.4.8 (Downward Löwenheim–Skolem Theorem). *Suppose that $\mathcal{L}$ is a countable language and $\mathfrak{B}$ is an $\mathcal{L}$-structure. Then $\mathfrak{B}$ has a countable elementary substructure.*

Proof. If B is finite or countably infinite, then $\mathfrak{B}$ is its own countable elementary substructure, so assume that B is uncountable. As the language $\mathcal{L}$ is countable, there are only countably many $\mathcal{L}$-formulas, and thus only countably many formulas of the form $\exists x\alpha$.

Let A_0 be any nonempty countable subset of B. We show how to build A_1 such that $A_0 \subseteq A_1$, and A_1 is countable. The idea is to add to A_0 witnesses for the truth (in $\mathfrak{B}$) of existential statements.

Notice that as A_0 is countable, there are only countably many functions $s' : \mathit{Vars} \to A_0$ that are eventually constant, by which we mean there is a natural number k such that if $i, j > k$, then $s'(v_i) = s'(v_j)$. (This is a nice exercise for those of you who have had a course in set theory or are reasonably comfortable with cardinality arguments.) Also, if we are given any ϕ and any $s : \mathit{Vars} \to A_0$, we can find an eventually constant $s' : \mathit{Vars} \to A_0$ such that s and s' agree on the free variables of ϕ, and thus $\mathfrak{B} \models \phi[s]$ if and only if $\mathfrak{B} \models \phi[s']$.

The construction of A_1: For each formula of the form $\exists x\alpha$ and each $s : \mathit{Vars} \to A_0$ such that $\mathfrak{B} \models \exists x\alpha[s]$, find an eventually constant $s' : \mathit{Vars} \to A_0$ such that s and s' agree on the free variables of $\exists x\alpha$. Pick an element $a_{\alpha,s'} \in B$ such that $\mathfrak{B} \models \alpha[s[x|a_{\alpha,s'}]]$, and let

$$A_1 = A_0 \cup \{a_{\alpha,s'}\}_{\text{all } \alpha, s: \mathit{Vars} \to A_0}.$$

Notice that A_1 is countable, as there are only countably many α's and countably many s'.

Continue this construction, iteratively building A_{n+1} from A_n. Let $A = \cup_{n=0}^{\infty} A_n$. As A is a countable union of countable sets, A is countable.

Now we have constructed a potential universe A for a substructure for $\mathfrak{B}$. We have to prove that A is closed under the functions of $\mathfrak{B}$ (by the remarks following Definition 3.4.1 this shows that $\mathfrak{A}$ is a substructure of $\mathfrak{B}$), and we have to show that $\mathfrak{A}$ satisfies the criteria set out in Lemma 3.4.7, so we will know that $\mathfrak{A}$ is an elementary substructure of $\mathfrak{B}$.

First, to show that A is closed under the functions of $\mathfrak{B}$, suppose that $a \in A$ and f is a unary function symbol (the general case is identical) and that $b = f^{\mathfrak{B}}(a)$. We must show that $b \in A$. Fix an n so large that $a \in A_n$, let ϕ be the formula $(\exists y)y = f(x)$, and let s be any assignment function into A such that $s(x) = a$. We know that $\mathfrak{B} \models (\exists y)y = f(x)[s]$, and we know that if $\mathfrak{B} \models \big(y = f(x)\big)[s[y|d]]$, then $d = b$. So, in our construction of A_{n+1} we must have used $a_{y=f(x),s} = b$, so $b \in A_{n+1}$, and $b \in A$, as needed.

In order to use Lemma 3.4.7, we must show that if α is a formula and $s : \mathit{Vars} \to A$ is such that $\mathfrak{B} \models \exists x\alpha[s]$, then there is an $a \in A$ such that $\mathfrak{B} \models \alpha[s[x|a]]$. So, fix such an α and such an s. Find an eventually constant $s' : \mathit{Vars} \to A$ such that s and s' agree on all the free variables of α. Thus $\mathfrak{B} \models \exists x\alpha[s']$, and all of the values of s' are elements of some fixed A_n, as s' takes on only a finite number of values. But then by construction of A_{n+1}, there is an element a of A_{n+1} such that $\mathfrak{B} \models \alpha[s'[x|a]]$. But this tells us (since s and s' agree on the free variables of α) that $\mathfrak{B} \models \alpha[s[x|a]]$, as needed.

So we have met the hypotheses of Lemma 3.4.7, and thus $\mathfrak{A}$ is a countable elementary substructure of $\mathfrak{B}$, as needed. □

> *Chaff:* We would like to look at a bit of this proof a little more closely. In the construction of A_1, what we did was to find an $a_{\alpha,s'}$ for each formula $\exists x\alpha$ and each $s : \mathit{Vars} \to A$, and the

point was that $a_{\alpha,s'}$ would be a witness to the truth in $\mathfrak{B}$ of the existential statement $\exists x\alpha$. So we have constructed a *function* which, given an existential formula $\exists x\alpha$ and an assignment function, finds a value for x that makes the formula α true. A function of this sort is called a **Skolem function**, and the construction of A in the proof of the Downward Löwenheim–Skolem Theorem can thus be summarized: Let A_0 be a countable subset of B, and form the closure of A_0 under the set of all Skolem functions. Then show that this closure is an elementary substructure of $\mathfrak{B}$.

Example 3.4.9. We saw an indication in Exercise 4 in Section 2.8.1 that the axioms of Zermelo–Fraenkel set theory (known as ZF) can be formalized in first-order logic. Accepting that as true (which it is), we know that if the axioms are consistent they have a model, and then by the Downward Löwenheim–Skolem Theorem, there must be a countable model for set theory. But this is interesting, as the following are all theorems of ZF:

- There is a countably infinite set.
- If a set a exists, then the collection of subsets of a exists.
- If a is countably infinite, then the collection of subsets of a is uncountable. (This is Cantor's Theorem).

Now, let us suppose that $\mathfrak{A}$ is our countable model of ZF, and suppose that a is an element of A and is countably infinite. If b is the set of all of the subsets of a, we know that b is uncountable (by Cantor's Theorem) and yet b must be countable, as all of the elements of b are in the model $\mathfrak{A}$, and $\mathfrak{A}$ is countable! So b must be both countable and uncountable! This is called (somewhat incorrectly) Skolem's paradox, and Exercise 8 asks you to figure out the solution to the paradox.

Probably the way to think about the Downward Löwenheim–Skolem Theorem is that it guarantees that if there are any infinite models of a given set of formulas, then there is a small (countably infinite means small) model of that set of formulas. It seems reasonable to ask if there is a similar guarantee about big models, and there is.

Proposition 3.4.10. *Suppose that Σ is a set of $\mathcal{L}$-formulas with an infinite model. If κ is an infinite cardinal, then there is a model of Σ of cardinality greater than or equal to κ.*

Proof. This is an easy application of the Compactness Theorem. Expand $\mathcal{L}$ to include κ new constant symbols c_i, and let $\Gamma = \Sigma \cup \{c_i \neq c_j \mid i \neq j\}$. Then Γ is finitely satisfiable, as we can take our given infinite model of Σ and interpret the c_i in that model in such a way that $c_i \neq c_j$ for any finite set of constant symbols. By the Compactness Theorem, there is a structure

$\mathfrak{A}$ that is a model of Γ, and thus certainly the cardinality of A is greater than or equal to κ. If we restrict $\mathfrak{A}$ to the original language, we get a model of Σ of the required cardinality. □

Corollary 3.4.11. *If Σ is a set of formulas from a countable language with an infinite model, and if κ is an infinite cardinal, then there is a model of Σ of cardinality κ.*

Proof. First, use Proposition 3.4.10 to get $\mathfrak{B}$, a model of Σ of cardinality greater than or equal to κ. Then, mimic the proof of the Downward Löwenheim–Skolem Theorem, starting with a set $A_0 \subseteq B$ of cardinality exactly κ. Then the A that is constructed in that proof also will have cardinality κ, and as $\mathfrak{A} \prec \mathfrak{B}$, $\mathfrak{A}$ will be a model of Σ of cardinality κ. □

Corollary 3.4.12. *If $\mathfrak{A}$ is an infinite $\mathcal{L}$-structure, then there is* no *set of first-order formulas that characterize $\mathfrak{A}$ up to isomorphism.*

Proof. More precisely, the corollary says that there is no set of formulas Σ such that $\mathfrak{B} \models \Sigma$ if and only if $\mathfrak{A} \cong \mathfrak{B}$. We know that there are models of Σ of all cardinalities, and we know that there are no bijections between sets of different cardinalities. So there must be many models of Σ that are not isomorphic to $\mathfrak{A}$. □

> *Chaff:* There are sets of axioms that do characterize infinite structures. For example, the second-order axioms of Peano Arithmetic include axioms to ensure that addition and multiplication behave normally, and they also include the principle of mathematical induction: If M is a set of numbers, if $0 \in M$, and if $S(n) \in M$ for every n such that $n \in M$, then $(\forall n)(n \in M)$.
>
> Any model of Peano Arithmetic is isomorphic to the natural numbers, but notice that we used two notions (sets of numbers and the elementhood relation) that are not part of our description of $\mathfrak{N}$. By introducing sets of numbers we have left the world of first-order logic and have entered second-order logic, and it is only by using second-order logic that we are able to characterize $\mathfrak{N}$. For a nice discussion of this topic, see [Bell and Machover 77, Chapter 7, Section 2].

The results from Proposition 3.4.10 to Corollary 3.4.12 give us models that are large, but they have a slightly different flavor from the Downward Löwenheim–Skolem Theorem, in that they do not guarantee that the small model is an elementary substructure of the large model. That is the content of the Upward Löwenheim–Skolem Theorem, a proof of which is outlined in the Exercises.

Theorem 3.4.13 (Upward Löwenheim–Skolem Theorem). *If $\mathcal{L}$ is a countable language, $\mathfrak{A}$ is an infinite $\mathcal{L}$-structure, and κ is a cardinal, then $\mathfrak{A}$ has an elementary extension $\mathfrak{B}$ such that the cardinality of B is greater than or equal to κ.*

3.4.1 Exercises

1. Suppose that $\mathfrak{B} \subseteq \mathfrak{A}$, that ϕ is of the form $(\forall x)\psi$, where ψ is quantifier-free, and that $\mathfrak{A} \models \phi$. Prove that $\mathfrak{B} \models \phi$. The short version of this fact is, "Universal sentences are preserved downward." Formulate and prove the corresponding fact for existential sentences.

2. Justify the *Chaff* following Definition 3.4.4.

3. Show that if $\mathfrak{A} \prec \mathfrak{B}$ and $\mathfrak{C} \prec \mathfrak{B}$ and $\mathfrak{A} \subseteq \mathfrak{C}$, then $\mathfrak{A} \prec \mathfrak{C}$.

4. Suppose that we have an **elementary chain**, a set of $\mathcal{L}$-structures such that
$$\mathfrak{A}_1 \prec \mathfrak{A}_2 \prec \mathfrak{A}_3 \prec \cdots$$
and let $\mathfrak{A} = \bigcup_{i=1}^{\infty} \mathfrak{A}_i$. So the universe A of $\mathfrak{A}$ is the union of the universes A_i, $R^{\mathfrak{A}} = \bigcup_{i=1}^{\infty} R^{\mathfrak{A}_i}$, etc. Show that $\mathfrak{A}_i \prec \mathfrak{A}$ for each i. [*Suggestion:* To show that $\mathfrak{A}_i \subseteq \mathfrak{A}$ is pretty easy by the definition. To get that $\mathfrak{A}$ is an *elementary* extension, you have to use induction on the complexity of formulas. Notice by the comments following Definition 3.4.4 that you need only prove one direction. You may find it easier to use $\exists$ rather than $\forall$ in the quantifier part of the inductive step of the proof.]

5. Prove Proposition 3.4.5.

6. Show that if $\mathfrak{A} \prec \mathfrak{B}$ and if there is an element $b \in B$ and a formula $\phi(x)$ such that $\mathfrak{B} \models \phi[s[x|b]]$ and for every other $\hat{b} \in B$, $\mathfrak{B} \not\models \phi[s[x|\hat{b}]]$, then $b \in A$. [*Suggestion:* This is very similar to Example 3.4.6.]

7. Suppose that $\mathfrak{B} = \{\mathbb{N}, +, \cdot\}$, and let $A_0 = \{2, 3\}$. Let F be the set of Skolem functions $\{f_{\alpha,s}\}$ corresponding to αs of the form $(\exists x)x = yz$. Find the closure of A_0 under F. [*Suggestion:* Do not forget that the assignment functions s that you need to consider are functions mapping into A_0 at first, then A_1, and so on. You probably want to explicitly write out A_1, then A_2, etc. We are using the notation here corresponding to the proof of Theorem 3.4.8.]

8. To say that a set a is countable means that there is a function with domain the natural numbers and codomain a that is a bijection. Notice that this is an existential statement, saying that a certain kind of function *exists*. Now, think about Example 3.4.9 and see if you can figure out why it is not really a contradiction that the set b is both countable and uncountable. In particular, think about what it means for an existential statement to be true in a structure $\mathfrak{A}$, as opposed to true in the real world (whatever *that* means!).

9. (Toward the Proof of the Upward Löwenheim–Skolem Theorem) If $\mathfrak{A}$ is an $\mathcal{L}$-structure, let $\mathcal{L}(A) = \mathcal{L} \cup \{\overline{a} \mid a \in A\}$, where each $\overline{a}$ is a new constant symbol. Then, let $\overline{\mathfrak{A}}$ be the $\mathcal{L}(A)$-structure having the same

universe as $\mathfrak{A}$ and the same interpretation of the symbols of $\mathcal{L}$ as $\mathfrak{A}$, and interpreting each $\overline{a}$ as a. Then we define the **complete diagram of** $\mathfrak{A}$ as

$$Th(\overline{\mathfrak{A}}) = \{\sigma \mid \sigma \text{ is an } \mathcal{L}(A)\text{-formula such that } \overline{\mathfrak{A}} \models \sigma\}.$$

Show that if $\overline{\mathfrak{B}}$ is any model of $Th(\overline{\mathfrak{A}})$, and if $\mathfrak{B} = \overline{\mathfrak{B}} \restriction_{\mathcal{L}}$, then $\mathfrak{A}$ is isomorphic to an elementary substructure of $\mathfrak{B}$. [*Suggestion:* Let $h : A \to B$ be given by $h(a) = \overline{a}^{\mathfrak{B}}$. Let C be the range of h. Show C is closed under $f^{\mathfrak{B}}$ for every f in $\mathcal{L}$, and thus C is the universe of $\mathfrak{C}$, a substructure of $\mathfrak{B}$. Then show h is an isomorphism between $\mathfrak{A}$ and $\mathfrak{C}$. Finally, show that $\mathfrak{C} \prec \mathfrak{B}$.]

10. Use Exercise 9 to prove the Upward Löwenheim–Skolem Theorem by finding a model $\overline{\mathfrak{B}}$ of the complete diagram of the given model $\mathfrak{A}$ such that the cardinality of $\overline{B}$ is greater than or equal to κ.

11. We can now fill in some of the details of our discussion of nonstandard analysis from Example 3.3.5. As the language $\mathcal{L}_{\mathbb{R}}$ of that example already includes constant symbols for each real number, the complete diagram of $\mathfrak{R}$ is nothing more than $Th(\mathfrak{R})$. Explain how Exercise 9 shows that there is an isomorphic copy of the real line living inside the structure $\mathfrak{A}$.

3.5 Summing Up, Looking Ahead

We have proven a couple of difficult theorems in this chapter, and by understanding the proof of the Completeness Theorem you have grasped an intricate argument with a wonderful idea at its core. Our results have been directed at structures: What kinds of structures exist? How can we (or can't we) characterize them? How large can they be?

The next chapter begins our discussion of Kurt Gödel's famous incompleteness theorems. Rather than discussing the strength of our deductive system as we have done in the last two chapters, we will now discuss the strength of sets of axioms. In particular, we will look at the question of how complicated a set of axioms must be in order to prove all of the true statements about the standard structure $\mathfrak{N}$.

In Chapter 4 we will introduce the idea of coding up the statements of $\mathcal{L}_{NT}$ as terms and will show that a certain set of nonlogical axioms is strong enough to prove some basic facts about the numbers coding up those statements. Then, in Chapters 5 and 6, we will bring those facts together to show that the expressive power we have gained has allowed us to express truths that are unprovable from our set of axioms.

Alternatively, after Chapter 4 you can move straight to Chapter 7 and approach the issue of provability from another direction. But for now, on to Chapter 4!

Chapter 4

Incompleteness From Two Points of View

4.1 Introduction

Now, we hope that you have been paying attention closely enough to be bothered by the title of this chapter. The preceding chapter was about completeness, and we proved the Completeness Theorem. Now we seem to be launching an investigation of incompleteness! This point is pretty confusing, so let us try to start out as clearly as possible.

In Chapter 3 we proved the completeness of our axiomatic system. We have shown that the deductive system described in Chapter 2 is sound and complete. What does this mean? For the collection of logical axioms and rules of inference that we have set out, any formula ϕ that can be deduced from a set of nonlogical axioms Σ will be true in all models of Σ under any variable assignment function (that's soundness), and furthermore any formula ϕ that is true in all models of Σ under every assignment function will be deducible from Σ (that's completeness). Thus, our deductive system is as nice as it can possibly be. The rough version of the Completeness and Soundness Theorems is: We can prove it if and only if it is true everywhere.

Now we will change our focus. Rather than discussing the wonderful qualities of our deductive system, we will concentrate on a particular language, $\mathcal{L}_{NT}$, and think about a particular structure, $\mathfrak{N}$, the natural numbers.

Wouldn't life be just great if we knew that we could prove every true statement about the natural numbers? Of course, the statements that we can prove depend on our choice of nonlogical axioms Σ, so let us start this paragraph over.

Wouldn't life be just great if we could find a set of nonlogical axioms

that could prove every true statement about the natural numbers? We would love to have a set of axioms Σ such that $\mathfrak{N} \models \Sigma$ (so our axioms are true statements about the natural numbers) and Σ is rich enough so that for every sentence σ, if $\mathfrak{N} \models \sigma$, then $\Sigma \vdash \sigma$. Since Σ has a model, we know that Σ is consistent, so by soundness our wished-for Σ will prove exactly those sentences that are true in $\mathfrak{N}$. The set of sentences of $\mathcal{L}_{NT}$ that are true in $\mathfrak{N}$ is called the **Theory of** $\mathfrak{N}$, or $Th(\mathfrak{N})$.

Since we know that a sentence is either true in $\mathfrak{N}$ or false in $\mathfrak{N}$, this set of axioms Σ is complete—complete in the sense that given any sentence σ, Σ will provide either a deduction of σ or a deduction of $\neg\sigma$.

Definition 4.1.1. A set of nonlogical axioms Σ in a language $\mathcal{L}$ is called **complete** if for every $\mathcal{L}$-sentence σ, either $\Sigma \vdash \sigma$ or $\Sigma \vdash \neg\sigma$.

> *Chaff:* To reiterate, in Chapter 3 we showed that our *deductive system* is complete. This means that for a given Σ, the deductive system will prove exactly those formulas that are logical consequences of Σ. When we say that a set of *axioms* is complete, we are saying that the axioms are strong enough to provide either a proof or a refutation of any sentence. This is harder.

Our goal is to find a complete and consistent set of $\mathcal{L}_{NT}$-axioms Σ such that $\mathfrak{N} \models \Sigma$. So this set Σ would be strong enough to prove every $\mathcal{L}_{NT}$-sentence that is true in the standard structure $\mathfrak{N}$. Such a set of axioms is said to axiomatize $Th(\mathfrak{N})$.

Definition 4.1.2. A set of axioms Σ is an **axiomatization of** $Th(\mathfrak{N})$ if for every sentence $\sigma \in Th(\mathfrak{N})$, $\Sigma \vdash \sigma$.

Actually, as stated, it is pretty easy to find an axiomatization of $Th(\mathfrak{N})$: Just let the axiom set be $Th(\mathfrak{N})$ itself. This set clearly axiomatizes itself, so we are finished! Off we go to have a drink. Of course, our answer to the search has the problem that we don't have an easy way to tell exactly which formulas are elements of the set of axioms. If we took a random sentence in $\mathcal{L}_{NT}$ and asked you if this sentence were true in the standard structure, we doubt you'd be able to tell us. The truth of nonrandom sentences is also hard to figure out—consider the twin prime conjecture, that there are infinitely many pairs of positive integers k and $k+2$ such that both k and $k+2$ are prime. People have been thinking about that one for over 2000 years and we don't know if it is true or not, although we seem to be currently (summer 2014) getting close. But at least as of now, we really have no idea if the twin prime conjecture is in $Th(\mathfrak{N})$ or not. So it looks like $Th(\mathfrak{N})$ is unsatisfactory as a set of nonlogical axioms.

So, to refine our question a bit, what we would like is a set of nonlogical axioms Σ that is simple enough so that we can recognize whether or not

a given formula is an axiom (so the set of axioms should be decidable) and strong enough to prove every formula in $Th(\mathfrak{N})$. So we search for a complete, consistent, decidable set of axioms for $\mathfrak{N}$. Unfortunately, our search is doomed to failure, and that fact is the content of Gödel's First Incompleteness Theorem, which we shall prove in Chapter 6 then again in Chapter 7. (You know a theorem is important if we're going to prove it more than once...)

What, precisely, is it that we will do? Given *any* complete, consistent, and decidable set of axioms for $\mathfrak{N}$, we are going to find a sentence σ that is a true statement about the natural numbers (so $\sigma \in Th(\mathfrak{N})$) but σ will not be provable from the collection of axioms. And how complicated must this sentence σ be? It turns out that it doesn't have to be very complicated at all. The next subsection will provide some structure to the collection of $\mathcal{L}_{NT}$-formulas and give us some language with which to talk about the complexity of formulas.

4.2 Complexity of Formulas

We work in the language of number theory

$$\mathcal{L}_{NT} = \{0, S, +, \cdot, E, <\},$$

and we will continue to work in this language for the next few chapters. $\mathfrak{N}$ is the standard model of the natural numbers,

$$\mathfrak{N} = (\mathbb{N}, 0, S, +, \cdot, E, <),$$

where the functions and relations are the usual functions and relations that you have known since you were knee high to a grasshopper. E is exponentiation, which will usually be written x^y rather than Exy or xEy.

One way to think about the simplest formulas of the language of the natural numbers (of any language, really) are the formulas that do not involve any quantifiers. It does seem natural that the formula $S0 = y$ is simpler than $\forall x S0 = y$. One baby step more complicated than quantifier-free formulas are the formulas that contain what we will call bounded quantifiers:

Definition 4.2.1. If x is a variable that does not occur in the term t, let us agree to use the following abbreviations:

$$(\forall x < t)\phi \quad \text{means} \quad \forall x(x < t \to \phi)$$

$$(\forall x \leq t)\phi \quad \text{means} \quad \forall x((x < t \vee x = t) \to \phi)$$

$$(\exists x < t)\phi \quad \text{means} \quad \exists x(x < t \wedge \phi)$$

$$(\exists x \leq t)\phi \quad \text{means} \quad \exists x((x < t \vee x = t) \wedge \phi).$$

These abbreviations will constitute the set of **bounded quantifiers**.

Thus, the formula $\exists x((\forall y < \overline{42})y = Sx)$ is a formula with one bounded quantifier and one unbounded quantifier.

Remember that our goal is to produce a sentence that is true in $\mathfrak{N}$ and not provable from our set of axioms. Might we be able to find such a formula that only contains bounded quantifiers? That would be really great, but unfortunately it is not going to happen. Recall our set of axioms N, back in Section 2.8? If you flip back to page 68 you will see that all of these axioms are true statements about the natural numbers, so they should be a consequence of any potential set of axioms for $Th(\mathfrak{N})$. But N is actually a pretty strong collection of statements. In particular, N is robust enough to prove every true statement about $\mathfrak{N}$ that contains only bounded quantifiers. Even better, N can refute every false statement that contains only bounded quantifiers. We'll prove this fact in Proposition 5.3.14. Since any potential candidate for an axiomitization of $\mathfrak{N}$ must be at least as strong as N, this tells us that our quest for a formula that is both true in $\mathfrak{N}$ and not provable must look at formulas that contain at least some unbounded quantifiers.

Definition 4.2.2. The collection of **Σ-formulas** is defined as the smallest set of $\mathcal{L}_{NT}$ formulas such that:

1. Every atomic formula is a Σ-formula.
2. Every negation of an atomic formula is a Σ-formula.
3. If α and β are Σ-formulas, then $\alpha \wedge \beta$ and $\alpha \vee \beta$ are both Σ-formulas.
4. If α is a Σ-formula, and x is a variable that does not occur in the term t, then the following are Σ-formulas: $(\forall x < t)\alpha$, $(\forall x \leq t)\alpha$, $(\exists x < t)\alpha$, $(\exists x \leq t)\alpha$.
5. If α is a Σ-formula and x is a variable, then $(\exists x)\alpha$ is a Σ-formula.

We will prove later (Theorem 5.3.13) that in fact our set of axioms N is strong enough to prove every true Σ-sentence, so even these formulas are not complicated enough to establish Gödel's incompleteness result. However, if instead of allowing an unbounded existential quantifier, we allow an unbounded universal quantifier, the situation is different.

Definition 4.2.3. The collection of **Π-formulas** is the smallest set of $\mathcal{L}_{NT}$-formulas such that:

1. Every atomic formula is a Π-formula.
2. Every negation of an atomic formula is a Π-formula.
3. If α and β are Π-formulas, then $\alpha \wedge \beta$ and $\alpha \vee \beta$ are both Π-formulas.

4. If α is a Π-formula, and x is a variable that does not occur in the term t, then the following are Π-formulas: $(\forall x < t)\alpha$, $(\forall x \leq t)\alpha$, $(\exists x < t)\alpha$, $(\exists x \leq t)\alpha$.

5. If α is a Π-formula and x is a variable, then $(\forall x)\alpha$ is a Π-formula.

So, while the set of Σ-formulas is closed under bounded quantification and unbounded existential quantification, the collection of Π-formulas is closed under bounded quantification and unbounded universal quantification.

The major result of the rest of the book, Gödel's First Incompleteness Theorem, states that if we are given any consistent and decidable set of axioms, then there will be a Π-formula σ such that σ is a true statement about the natural numbers but there is no deduction from our axioms of the formula σ. So our set of axioms must be incomplete. Getting to that theorem will occupy us for the rest of our time together.

You might notice that every denial of a Σ-formula is logically equivalent to a Π-formula, and vice versa (see Exercise 3). If we take the intersection of the collection of Σ-formulas and the collection of Π-formulas, we have the Δ-formulas:

Definition 4.2.4. The collection of **Δ-formulas** is the intersection of the collection of Σ-formulas with the set of Π-formulas.

Thus in every Δ-formula all quantifiers are bounded. It will turn out that our mysterious (well, it isn't really that mysterious) set of axioms N is strong enough to prove every true-in-$\mathfrak{N}$ Δ-formula and refute every false-in-$\mathfrak{N}$ Δ-formula. This will be very important to us.

4.2.1 Exercises

1. Referring to Definition 4.2.2, explain in detail why the following formulas are (or are not) Σ-formulas.

   (a) $S0 + S0 = SS0$

   (b) $\neg(0 < 0 \vee 0 < S0)$

   (c) $(\forall x < \overline{17})x < \overline{17}$

   (d) $S0 \cdot S0 = S0 \wedge (\exists y < x)(\exists z < y)y + z = x$

   (e) $(\forall y)(y < 0 \rightarrow 0 = 0)$

   (f) $(\exists x)(x < x)$

2. Let's define the set of Cool Formulas to be the smallest set of $\mathcal{L}_{NT}$-formulas that:

   (a) Contains all atomic formulas.

(b) Contains all negations of atomic formulas.

(c) Is closed under the connectives $\wedge$ and $\vee$.

(d) Is closed under bounded quantifiers and the quantifier $\exists$.

Prove that a formula is Cool if and only if the formula is a Σ-formula. (The four conditions above are sometimes used to define the set of Σ-formulas. You've just proved that the definition here is equivalent to Definition 4.2.2.)

3. Think about the Σ-formula

$$\alpha \text{ is } x < y \vee (\forall z < w)x + \overline{17} = \overline{42}.$$

(a) Is α a Π-formula?

(b) Is $\neg\alpha$ a Π-formula?

(c) Can you find a Π-formula that is equivalent to $\neg\alpha$?

(d) Carefully prove that, if α is any Σ-formula, then $\neg\alpha$ is logically equivalent to a Π-formula.

4.3 The Roadmap to Incompleteness

At the end of the day, we will want to be looking at this true-in-$\mathfrak{N}$ Π-formula σ that we have constructed and be able to say to it, "There is no deduction of you." The construction of the formula σ will involve rather detailed analysis of the collections of deductions, so it will be convenient, if initially messy, to have a way to translate deductions into natural numbers. Thus for example, rather than saying "This long sequence of formulas is a deduction of the formula $0 = 1$," we could say "The number 42 is a code for a deduction of $0 = 1$."

The other advantage of having this coding is that it will allow us to code up statements about numbers that code up statements. For example, it might be the case that the number 24601 is a code for the statement, "The number 42 is a code for a deduction of $0 = 1$." Or even (and this is the key idea) 24601 might be the code for the statement, "There is no number that is a code for a deduction of the formula for which I am the code." The rather messy details of this will be covered in Chapter 6.

This all hinges on the facts that it is easy to code statements as numbers and decode numbers to see what statements they encode. Also, it is easy to check whether a potential deduction is, in fact, a deduction. Recall that a deduction is nothing more than a finite sequence of formulas, each one of which is either an axiom or follows from previous formulas in the sequence via a rule of inference. Thus, once we decide on a way to code formulas and to code sequences of formulas, it will not be a problem to examine a number and decide if that number codes up a deduction or not. Thus our

path forward will be to fix our coding scheme, prove that the coding is nice, use the coding scheme in order to construct the formula σ, and then prove that σ is both true and not provable. This route to the Incompleteness Theorem is followed in Chapters 5 and 6.

4.4 An Alternate Route

A second route to incompleteness focuses not on formulas and deductions, but rather on functions mapping the natural numbers to the natural numbers. This line of reasoning, developed in the late 1930s, has a clear connection to computation and theoretical computer science. By thinking carefully about what it means for a function to be computable and just what a computation *is*, we will once again be able to show the existence of a formula that is true and yet not provable. The details of this plan are laid out in Chapter 7.

When is a function computable? The idea is that to say that a function f is computable on input k means that there is a sequence of easy steps that leads to the correct output $f(k)$. We will make this precise in Chapter 7, but roughly it means that one can start with some easy functions and build up the function f by some relatively simple operations on previously defined functions.

Thus it looks like that in order to carefully define what it means to compute a function, we will be required to discuss sequences of partial computations, and once again it will be convenient to be able to code up these sequences as natural numbers. So even if we take this functionally based route to the Incompleteness Theorem, we will need to be familiar with some coding apparatus. Since both of our routes to the Incompleteness Theorem will require us to code up sequences, we will take the rest of this chapter to fix our notation and establish a couple of easy results.

4.5 How to Code a Sequence of Numbers

Suppose we have a finite sequence of numbers, maybe

$$2, 4, 3, 5, 9$$

and we wish to code them up as a single number. An easy way to do this would be to code the sequence into the exponents of the first few prime numbers and then multiply them together:

$$2^2 \cdot 3^4 \cdot 5^3 \cdot 7^5 \cdot 11^9 = 1605016087126798500.$$

This would be easy, but unfortunately it will not suffice for our purposes, so we'll have to be a little sneakier. Fortunately, by being clever now, life will be simpler later, so it seems to be worth the effort.

You're probably thinking that it would be easy to decide if a number was a code for a sequence. Obviously $72 = 2^3 3^2$ wants to be the code for the sequence $3, 2$, and the number 10 is not a code number, since $10 = 2 \cdot 5$ is not a product of the first few primes.

Sorry.

Your perfectly fine idea runs into trouble if we try to code up sequences that include the number 0. For example if we were to code up the sequence

$$1, 0, 1$$

we would get $2^1 \cdot 3^0 \cdot 5^1 = 10$, and so 10 should be a code number. But your idea about coding things as exponents really was a good one, and we can save it if we just agree that whenever we wish to code a finite sequence of numbers $a_1, a_2, \ldots, a_k$, we will use the exponents $a_1 + 1, a_2 + 1, \ldots, a_k + 1$, which takes care of those pesky 0's. Furthermore, when we decode we will automatically subtract one from every exponent, so you'll never have to think about it. (Hey, that's why we, the authors, are paid the big bucks!)

The idea here is that a sequence of k natural numbers should be coded by a product of the first k primes raised to non-zero powers. So the empty sequence will, naturally, be coded by a product of the first 0 primes raised to some power. In other words, the code of the empty sequence will be the number 1. Let us make this more formal:

Definition 4.5.1. The function p is the function mapping the natural numbers to the natural numbers, where $p(0) = 1$ and $p(k)$ is the k^{th} prime for $k \geq 1$. Thus $p(0) = 1$, $p(1) = 2$, $p(2) = 3$, and so on. We will often write p_i instead of $p(i)$.

Definition 4.5.2. Let $\mathbb{N}^{<\mathbb{N}}$ denote the set of finite sequences of natural numbers.

Definition 4.5.3. We define the coding function $\langle\cdot\rangle : \mathbb{N}^{<\mathbb{N}} \to \mathbb{N}$ by

$$\langle (a_1, a_2, \ldots, a_k) \rangle = \begin{cases} 1 & \text{if } k = 0 \\ \prod_{i=1}^{k} p_i^{a_i+1} & \text{if } k > 0 \end{cases}$$

where p_i is the ith prime number.

We will write $\langle a_1, a_2, \ldots, a_k \rangle$ rather than $\langle (a_1, a_2, \ldots, a_k) \rangle$.

It would be pointless to be able to code up sequences without being able to decode them, and the next functions that we define will let us do that. But before getting there, we need to acknowledge that we will be depending on the Fundamental Theorem of Arithmetic, which states that

every positive integer greater than one can be expressed in exactly one way (up to order) as a product of primes. The proof of this theorem (first proven by Euclid) is nontrivial and beyond the scope of this book, but is certainly worth looking up. But we will happily remember that the theorem is true, and use it freely.

There is, however, a messy detail with which we have to deal, and we might as well deal with it now.

Our decoding functions will have to be total functions, by which we mean that each of the functions will have domain $\mathbb{N}$. But lots of natural numbers are not the code of sequences, and we have to figure out how to deal with such numbers. To make the definitions that are coming up reasonable, and to save us more difficulties later, we start by defining the set of numbers that are codes.

Definition 4.5.4. Let $C = \{a \in \mathbb{N} \mid \left(\exists s \in \mathbb{N}^{<\mathbb{N}}\right)\, a = \langle s \rangle\}$. We will call C the set of *code numbers*.

Notice that it is easy to check whether or not $a \in C$. All we need to do is factor a and see if either $a = 1$ or if a is a product of the first few primes.

Now we can get along to uncoding:

Definition 4.5.5. The function $|\cdot| : \mathbb{N} \to \mathbb{N}$ is defined by

$$|a| = \begin{cases} k & \text{if } a \in C \text{ and } a = \langle a_1, a_2, \ldots, a_k \rangle \\ 0 & \text{otherwise.} \end{cases}$$

If a is a code number, we will say that $|a|$ is the *length of* a.

Chaff: Ok, where did we use the Fundamental Theorem of Arithmetic?

Notice that we have defined the function $|\cdot|$ in such a way that its domain is the entire set of natural numbers. Since lots of natural numbers will not be codes of finite sequences, we have had to make a choice about how we would define our length function on those numbers. So, by definition, if a is any number that is not of the form $p_1^{a_1+1} p_2^{a_2+1} \ldots p_k^{a_k+1}$, then $|a| = 0$. But we will not talk about the length of such a number.

Please be careful about the difference between $|a|$ and $|\langle a \rangle|$. See Exercise 2.

Definition 4.5.6. For each $i \in \mathbb{N}$ with $i \geq 1$, let $(\cdot)_i$ be the function with domain $\mathbb{N}$ and codomain $\mathbb{N}$ defined by

$$(a)_i = \begin{cases} a_i & \text{if } a \in C \text{ and } a = \langle a_1, a_2, \ldots, a_k \rangle \text{ and } 1 \leq i \leq k \\ 0 & \text{otherwise.} \end{cases}$$

The skeptical (and sharp-eyed) reader will have noticed another little detail here. Consider the number $a = 10$. We have agreed that 10 does not code up a sequence, and thus Definition 4.5.5 tells us that $|10| = 0$. However, if we use our decoding function that we have just defined, maybe looking for the seventh term of the sequence, and we plug in the input 10, we find $(10)_7 = 0$. This is slightly annoying, since the casual observer might think that 10 is supposed to code a sequence of length 0, but beyond aggravating the authors, this side effect will not bother us at all.

We will also need to be able to put the codes of two sequences together, one after the other, and the next function allows us to do so.

Definition 4.5.7. The function $^\frown : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ is defined by

$$a^\frown b = \begin{cases} \langle a_1, \ldots, a_k, b_1, \ldots, b_l \rangle & \text{if } a = \langle a_1, \ldots, a_k \rangle \text{ and } b = \langle b_1, \ldots, b_l \rangle, \\ & \quad \text{with } (a_1, \ldots, a_k) \in \mathbb{N}^{<\mathbb{N}} \text{ and} \\ & \quad (b_1, \ldots, b_l) \in \mathbb{N}^{<\mathbb{N}} \\ 0 & \text{otherwise.} \end{cases}$$

It might be worthwhile to work through an example at this point. Suppose we wished to compute $793800^\frown 73500$. We would first do a lot of factoring to find that $793800 = 2^3 3^4 5^2 7^2$, so $793800 = \langle 2, 3, 1, 1 \rangle$. Similarly, $73500 = \langle 1, 0, 2, 1 \rangle$. So, by definition

$$\begin{aligned} 793800^\frown 73500 = \langle 2, 3, 1, 1, 1, 0, 2, 1 \rangle &= 2^3 3^4 5^2 7^2 11^2 13^1 17^3 19^2 \\ &= 2214592288108200, \end{aligned}$$

while $793801^\frown 73500 = 0$.

The functions introduced above allow us to code finite sequences and, given a code number a, decode it. In other words, if $a \in C$ and $a = \langle a_1, \ldots, a_k \rangle$, then for each i such that $1 \leq i \leq |a|$, $(a)_i = a_i$.

At this point, we have built all of the coding apparatus that we will need. Whether we approach incompleteness through formulas or through computations, we will be able to code and decode the objects and sequences of the objects. We have built the infrastructure. In Chapters 5 and 6 we will apply the coding to formulas, while in Chapter 7 we use the coding on computations. Both routes lead us to incompleteness.

4.5.1 Exercises

1. Compute the following:
   (a) $\langle 3, 0, 4, 2, 1 \rangle$
   (b) $(16910355000)_3$
   (c) $|16910355000|$
   (d) $(16910355000)_{42}$

(e) $\langle 2,7,1,8\rangle^\frown\langle 2,8,1\rangle$

(f) $17^\frown 42$

2. Find a number a such that $|a| \neq |\langle a\rangle|$ or prove that no such a exists. Then find a number b such that $|b| = |\langle b\rangle|$ or prove that no such b exists.

4.6 An Old Friend

Back in Example 2.8.3 we introduced the collection of nonlogical axioms N. Just because they are so important, we'll reprint them here:

The Axioms of N

1. $(\forall x)\neg Sx = 0$.
2. $(\forall x)(\forall y)\big[Sx = Sy \to x = y\big]$.
3. $(\forall x)x + 0 = x$.
4. $(\forall x)(\forall y)x + Sy = S(x + y)$.
5. $(\forall x)x \cdot 0 = 0$.
6. $(\forall x)(\forall y)x \cdot Sy = (x \cdot y) + x$.
7. $(\forall x)xE0 = S0$.
8. $(\forall x)(\forall y)xE(Sy) = (xEy) \cdot x$.
9. $(\forall x)\neg x < 0$.
10. $(\forall x)(\forall y)\big[x < Sy \leftrightarrow (x < y \vee x = y)\big]$.
11. $(\forall x)(\forall y)\big[(x < y) \vee (x = y) \vee (y < x)\big]$.

At that time (Lemma 2.8.4) we proved some things about the strength of this innocuous-looking set of axioms, for example, if the natural number a is equal to the sum $b + c$, then $N \vdash \overline{a} = \overline{b} + \overline{c}$. We will need some further results about the strength of N that will be proven in detail in Chapter 5. We state them here and give some examples.

Recall that we have defined the collections of Δ-formulas as the formulas in the language of number theory that contain no unbounded quantifiers. For a specific example, consider $\phi(x)$, the formula with one free variable

$$\phi(x) :\equiv (\exists y \leq x)\overline{2}y = x$$

which states that x is an even number. Suppose that we consider two different sentences associated with ϕ: $\phi(\overline{2})$ and $\phi(\overline{3})$. We will all agree that $\phi(\overline{2})$ is a true statement about $\mathfrak{N}$, while $\phi(\overline{3})$ is false. What is so wonderful

about our set of non-logical axioms N is that N is strong enough to prove the first statement and refute the second:

$$N \vdash \phi(\overline{2}) \qquad\qquad N \vdash \neg\phi(\overline{3})$$

This is a general fact about the relation between Δ-formulas and N, which we will state here and prove as Proposition 5.3.14:

Proposition 4.6.1. *If $\phi(\underset{\sim}{x})$ is a Δ-formula with free variables $\underset{\sim}{x}$, if $\underset{\sim}{t}$ are variable-free terms, and if $\mathfrak{N} \models \phi(\underset{\sim}{t})$, then $N \vdash \phi(\underset{\sim}{t})$. If, on the other hand, $\mathfrak{N} \models \neg\phi(\underset{\sim}{t})$, then $N \vdash \neg\phi(\underset{\sim}{t})$.*

If we allow an unbounded existential quantifier, the situation changes slightly. Our set of axioms N is strong enough to prove the true Σ-sentences, but it cannot refute the false Σ-sentences. This result, called Σ-*completeness*, will be proven as Proposition 5.3.13:

Proposition 4.6.2. *If $\phi(\underset{\sim}{x})$ is a Σ-formula with free variables $\underset{\sim}{x}$, if $\underset{\sim}{t}$ are variable-free terms, and if $\mathfrak{N} \models \phi(\underset{\sim}{t})$, then $N \vdash \phi(\underset{\sim}{t})$.*

At one level, this should not be too surprising, given that N is strong enough to prove or refute sentences that have no unbounded quantifiers. Suppose that ϕ is a true Σ-sentence. Then ϕ looks (roughly) like $\exists x\psi(x)$, where ψ has one free variable. Since $\exists x\psi(x)$ is true in $\mathfrak{N}$, then there is some natural number k such that $\psi(k)$ is true. But then $\psi(\overline{k})$ is a true-in-$\mathfrak{N}$ sentence with no unbounded quantifiers. By assumption, N is strong enough to prove $\psi(\overline{k})$, and so by the usual rules of logic, N also proves $\exists x\psi(x)$; i.e., N proves ϕ.

On the other hand, if our Σ-sentence ϕ is false in $\mathfrak{N}$, that just means that there is no natural number k such that $\psi(k)$ is true. Now, since $\psi(\overline{k})$ has no unbounded quantifiers, by assumption that means that $N \vdash \neg\psi(\overline{k})$ for every natural number k. But we have already seen that there are lots of structures of non-standard arithmetic where a property can be false of every natural number but still true of some other element of the universe. So just because N can prove that ψ is false of every natural number, there is no reason *a priori* to assume that N can then prove that ψ is false of *everything*. And in fact, it cannot.

So, the short version:

N is strong enough to prove true Σ-sentences, but not strong enough to refute false Σ-sentences.

Equivalently, since the denial of a Σ-sentence is equivalent to a Π-sentence:

N is strong enough to prove every true Σ-sentence, but not strong enough to prove every true Π-sentence.

For example, consider the Goldbach Conjecture, which states that every even number greater than two can be written as the sum of two primes. It is not difficult to see that the Goldbach Conjecture can be written formally as a Π-sentence, but unfortunately we currently do not know whether the Goldbach Conjecture is true or not. Suppose for a second that the conjecture was false. Then its denial, equivalent to a Σ-sentence, would be true, and therefore N would be able to prove the denial. Not surprising, really. All we would have to do is find an even number that is a counterexample to the Goldbach Conjecture and check that it isn't the sum of two primes. (Warning: If you're going to start looking for the counterexample, go big or go home. The conjecture has been verified for all even numbers up to at least 4×10^{18}.)

On the other hand, if the Goldbach Conjecture is true, then there is no reason to believe that N is strong enough to prove that fact. Which, by the way, means that if we could prove that N is not strong enough to decide the Goldbach Conjecture, then the Goldbach Conjecture is true!

4.7 Summing Up, Looking Ahead

The big concepts that we have introduced in this chapter are three. First is coding, both the idea behind it and the mechanism that we will use to accomplish it. Secondly, we have defined what it means to talk about the complexity of a formula, and introduced the collections of Σ-, Π-, and Δ-formulas. Then, we reintroduced the collection of axioms N, and we mentioned (but did not prove) that N is Σ-complete; N is strong enough to prove all Σ-sentences that are true in $\mathfrak{N}$.

Before us we have the path to Gödel's Incompleteness Theorem. But we should say "paths" rather than path. You, the reader, get to choose what happens next. If you would like to see a development of incompleteness that is based on formulas in $\mathcal{L}_{NT}$, then continue on into Chapter 5. If, on the other hand, you are more interested in an argument that focuses on computations rather than formulas, skip Chapters 5 and 6 for now and move on to Chapter 7. Of course, on a second reading you should look over (at least briefly) the material that you skipped; there are insights and subtleties to be appreciated in each approach! We will bring things back together in Chapter 8 and point you toward further reading in Mathematical Logic that will introduce you to further results and other areas of study in this fascinating field. But first, on to Incompleteness!

Chapter 5

Syntactic Incompleteness— Groundwork

5.1 Introduction

There is a fair bit of groundwork to cover before we get to the Incompleteness Theorem, and much of that groundwork is rather technical. Here is a thumbnail sketch of our plan to reach the theorem: The proof of the First Incompleteness Theorem essentially consists of constructing a certain sentence θ and noticing that θ is, by its very nature, a true statement in $\mathfrak{N}$ and a statement that is unprovable from our axioms. So the groundwork consists of making sure that this yet-to-be-constructed θ exists and does what it is supposed to do. In this chapter we will specify our language and reintroduce N, a set of nonlogical axioms. The axioms of N will be true sentences in $\mathfrak{N}$. We will show that N, although very weak, is strong enough to prove some crucial results. We will then show that our language is rich enough to express several ideas that will be crucial in the construction of θ.

In Chapter 6 we will prove Gödel's Self-Reference Lemma and use that lemma to construct the sentence θ. We shall then state and prove the First Incompleteness Theorem, that there can be no decidable, consistent, complete set of axioms for $\mathfrak{N}$. We will finish the chapter with a discussion of Gödel's Second Incompleteness Theorem, which shows that no reasonably strong set of axioms can ever hope to prove its own consistency.

5.2 The Language, the Structure, and the Axioms of N

We work in the language of number theory

$$\mathcal{L}_{NT} = \{0, S, +, \cdot, E, <\},$$

and we will continue to work in this language for the next two chapters. $\mathfrak{N}$ is the standard model of the natural numbers,

$$\mathfrak{N} = \langle \mathbb{N}, 0, S, +, \cdot, E, < \rangle,$$

where the functions and relations are the standard functions and relations on the natural numbers.

We will now establish a set of nonlogical axioms, N. You will notice that the axioms are clearly sentences that are true in the standard structure, and thus if T is *any* set of axioms such that $T \vdash \sigma$ for all σ such that $\mathfrak{N} \models \sigma$, then $T \vdash N$. So, as we prove that several sorts of formulas are derivable from N, remember that those same formulas are also derivable from any set of axioms that has any hope of providing an axiomatization of the natural numbers.

The axiom system N was introduced in Example 2.8.3 and is reproduced on the next page. These 11 axioms establish some of the basic facts about the successor function, addition, multiplication, exponentiation, and the $<$ ordering on the natural numbers.

> *Chaff:* To be honest, the symbol E and the axioms about exponentiation are not needed here. It is possible to do everything that we do in the next couple of chapters by defining exponentiation in terms of multiplication, and introducing E as an abbreviation in the language. This has the advantage of showing more explicitly how little you need to prove the incompleteness theorems, but adds some complications to the exposition. We have decided to introduce exponentiation explicitly and add a couple of axioms, which will allow us to move a little more cleanly through the proofs of our theorems.

The Axioms of N

1. $(\forall x)\neg Sx = 0$.
2. $(\forall x)(\forall y)\big[Sx = Sy \to x = y\big]$.
3. $(\forall x)x + 0 = x$.
4. $(\forall x)(\forall y)x + Sy = S(x + y)$.
5. $(\forall x)x \cdot 0 = 0$.
6. $(\forall x)(\forall y)x \cdot Sy = (x \cdot y) + x$.
7. $(\forall x)xE0 = S0$.
8. $(\forall x)(\forall y)xE(Sy) = (xEy) \cdot x$.
9. $(\forall x)\neg x < 0$.
10. $(\forall x)(\forall y)\big[x < Sy \leftrightarrow (x < y \vee x = y)\big]$.
11. $(\forall x)(\forall y)\big[(x < y) \vee (x = y) \vee (y < x)\big]$.

5.2.1 Exercises

1. You have already seen that N is not strong enough to prove the commutative law of addition (Exercise 8 in Section 2.8.1). Use this to show that N is not complete by showing that

$$N \nvdash (\forall x)(\forall y)x + y = y + x$$

and

$$N \nvdash \neg\big[(\forall x)(\forall y)x + y = y + x\big].$$

2. Suppose that Σ provides an axiomatization of $Th(\mathfrak{N})$. Suppose σ is a formula such that $N \vdash \sigma$. Show that $\Sigma \vdash \sigma$.

3. Suppose that $\mathfrak{A}$ is a nonstandard model of arithmetic. If $Th(\mathfrak{A})$ is the collection of sentences that are true in $\mathfrak{A}$, is $Th(\mathfrak{A})$ complete? Does $Th(\mathfrak{A})$ provide an axiomatization of $\mathfrak{N}$? Of $\mathfrak{A}$?

5.3 Representable Sets and Functions

For the sake of discussion, suppose that we let $f(x) = x^2$. It will not surprise you to find out that $f(4) = 16$, so we would like to write $\mathfrak{N} \models f(4) = 16$. Unfortunately, we are not allowed to do this, since the symbol f, not to mention 4 and 16, are not part of the language.

What we can do, however, is to represent the function f by a formula in $\mathcal{L}_{NT}$. To be specific, suppose that $\phi(x, y)$ is

$$y = ExSS0.$$

Then, if we allow ourselves once again to use the abbreviation $\overline{a}$ for the $\mathcal{L}_{NT}$-term $\underbrace{SSS\cdots S}_{a S\text{'s}}0$, we can assert that

$$\mathfrak{N} \models \phi(\overline{4}, \overline{16})$$

which is the same thing as

$$\mathfrak{N} \models = SSSSSSSSSSSSSSSS0ESSSS0SS0.$$

(Aren't you glad we don't use the official language very often?) Anyway, the situation is even better than this, for $\phi(\overline{4}, \overline{16})$ is derivable from N rather than just true in $\mathfrak{N}$. In fact, if you look back at Lemma 2.8.4, you probably won't have any trouble believing the following statements:

- $N \vdash \phi(\overline{4}, \overline{16})$
- $N \vdash \neg\phi(\overline{4}, \overline{17})$
- $N \vdash \neg\phi(\overline{1}, \overline{714})$

In fact, this formula ϕ is such, and N is such, that if a is any natural number and $b = f(a)$, then

$$N \vdash \forall y\big[\phi(\overline{a}, y) \leftrightarrow y = \overline{b}\big].$$

We will say that the formula ϕ represents the function f in the theory N.

Definition 5.3.1. A set $A \subseteq \mathbb{N}^k$ is said to be **representable** (in N) if there is an $\mathcal{L}_{NT}$-formula $\phi(\underset{\sim}{x})$ such that

$$\begin{aligned} &\forall \underset{\sim}{a} \in A \quad && N \vdash \phi(\overline{\underset{\sim}{a}}) \\ &\forall \underset{\sim}{b} \notin A \quad && N \vdash \neg\phi(\overline{\underset{\sim}{b}}). \end{aligned}$$

In this case we will say that the formula ϕ **represents** the set A.

Definition 5.3.2. A set $A \subseteq \mathbb{N}^k$ is said to be **weakly representable** (in N) if there is an $\mathcal{L}_{NT}$-formula $\phi(\underset{\sim}{x})$ such that

$$\begin{aligned} &\forall \underset{\sim}{a} \in A \quad && N \vdash \phi(\overline{\underset{\sim}{a}}) \\ &\forall \underset{\sim}{b} \notin A \quad && N \nvdash \phi(\overline{\underset{\sim}{b}}). \end{aligned}$$

In this case we will say that the formula ϕ **weakly represents** the set A.

Notice that if A is representable, then A is weakly representable. On a few occasions we will talk about a set being representable in T, where T is a different set of $\mathcal{L}_{NT}$-formulas. This just means that all of the deductions mentioned in the previous two definitions should be deductions-from-T, rather than deductions-from-N.

> *Chaff:* A bit of notation has slipped in here. Rather than writing $x_1, x_2, \ldots, x_k$ over and over and over again in the next few sections, we will abbreviate this as $\underset{\sim}{x}$. Similarly, $\overline{\underset{\sim}{x}}$ is shorthand for $\overline{x_1}, \overline{x_2}, \ldots, \overline{x_k}$. If you want, you can just assume that there is only one x—it won't make any difference to the exposition.

We will also discuss representable functions, and this brings up a couple of subtle points that need to be addressed. The general idea is that a function $f : \mathbb{N}^k \to \mathbb{N}$ should be representable if N is able to prove that a formula that represents the function does the right thing, but historically the term weakly representable has been applied to functions whose domains are (possibly strict) subsets of $\mathbb{N}^k$, which adds some complexity.

To begin with, we will need to be able to talk about the domains of functions with a little more precision.

Definition 5.3.3. Suppose that $A \subseteq \mathbb{N}^k$ and suppose that $f : A \to \mathbb{N}$. If $A = \mathbb{N}^k$ we will say that f is a **total function**. If $A \subsetneq \mathbb{N}^k$, we will call f a **partial function**.

Now we can define what it means for a function to be representable or weakly representable.

Definition 5.3.4. Suppose that $f : \mathbb{N}^k \to \mathbb{N}$ is a total function. We will say that f is a **representable function** (in N) if there is an $\mathcal{L}_{NT}$ formula $\phi(x_1, \ldots, x_{k+1})$ such that, for all $a_1, a_2, \ldots a_{k+1} \in \mathbb{N}$,

$$\text{If } f(a_1, \ldots, a_k) = a_{k+1}, \text{ then } N \vdash \phi(\overline{a_1}, \ldots, \overline{a_{k+1}})$$
$$\text{If } f(a_1, \ldots, a_k) \neq a_{k+1}, \text{ then } N \vdash \neg\phi(\overline{a_1}, \ldots, \overline{a_{k+1}}).$$

Notice that a total function is a representable function if and only if it is a representable subset of $\mathbb{N}^{k+1}$. See Exercise 4.

Definition 5.3.5. Suppose that $A \subseteq \mathbb{N}^k$ and $f : A \to \mathbb{N}$ is a (possibly) partial function. We will say that f is a **weakly representable function** (in N) if there is an $\mathcal{L}_{NT}$ formula $\phi(x_1, \ldots, x_{k+1})$ such that, for all $a_1, a_2, \ldots a_{k+1} \in \mathbb{N}$,

$$\text{If } f(a_1, \ldots, a_k) = a_{k+1}, \text{ then } N \vdash \phi(\overline{a_1}, \ldots, \overline{a_{k+1}})$$
$$\text{If } f(a_1, \ldots, a_k) \neq a_{k+1}, \text{ then } N \nvdash \phi(\overline{a_1}, \ldots, \overline{a_{k+1}}).$$

Chaff: In the definition above, notice that if, for example, 5 is not an element of the domain of the unary function f, then $f(5)$ is not going to be equal to anything, so we know that $f(5) \neq 17$. All we ask, in that case, is that N does not prove $\phi(\overline{5}, \overline{17})$. We do not need, and cannot expect, N to prove $\neg\phi(\overline{5}, \overline{17})$.

How important is it to know whether a function is total or not? With regards to representability, the big result is the following, the proof of which is omitted.

Proposition 5.3.6. *Suppose that f is a total function from $\mathbb{N}^k$ to $\mathbb{N}$. Then f is representable if and only if f is weakly representable.*

Partial functions will be very important to us as we work through Chapter 7, but for the next couple of chapters, almost all of our functions will be total. If f is representable, we can say a little more about what N can prove about f.

Proposition 5.3.7. *Suppose that $f : \mathbb{N}^k \to \mathbb{N}$ is a total function. Then the following are equivalent.*

1. *f is a representable function.*
2. *There exists an $\mathcal{L}_{NT}$-formula $\psi(x_1, \ldots, x_{k+1})$ such that for all $\underset{\sim}{a} \in \mathbb{N}^k$,*

$$N \vdash (\forall y)\big[\psi(\underset{\sim}{\overline{a}}, y) \leftrightarrow y = \overline{f(\underset{\sim}{a})}\big].$$

Proof. Exercise 9 provides an outline of a proof. □

Chaff:

What's in a name? that which we call a rose
By any other name would smell as sweet;
So Romeo would, were he not Romeo call'd,
Retain that dear perfection which he owes
Without that title.

—*Romeo and Juliet,* Act II, Scene ii

In computer science courses, in many mathematical logic texts, and in fact in Chapter 7 of this book, a different approach is taken when representable sets and functions are introduced. Starting with certain initial functions, the idea of recursion, and an object called the μ-operator, a collection of partial functions is defined such that each function in the collection is effectively calculable. This is called the collection of computable functions, which leads to something called decidable (or computable) sets. Then these texts prove that the collection of representable sets

that we just defined is the same as the collection of decidable sets. Thus we all end up at the same place, with nicely defined collection of sets and functions, that we call computable. Or representable. Or recursive. So if you are confused by the different definitions, just remember that they all define the same concept, and remember that the objects that are recursive (or representable or computable) are (in some sense) the simple ones, the ones where membership can be proved in N.

To be fair, it is not quite as simple as Juliet makes it out to be. (It never is, is it?) The path that we have taken to representable sets is clean and direct but emphasizes the deductions over the functions. The approach through initial functions stresses the fact that everything that we discuss can be calculated, and that viewpoint gives a natural tie between the logic that we have been discussing and its applications to computer science. For more on this connection, see Section 5.4 and Chapter 7.

Definition 5.3.8. We will say that a set $A \subseteq \mathbb{N}^k$ is **definable** if there is a formula $\phi(\underset{\sim}{x})$ such that

$$\begin{array}{ll} \forall \underset{\sim}{a} \in A & \mathfrak{N} \models \phi(\overline{\underset{\sim}{a}}) \\ \forall \underset{\sim}{b} \notin A & \mathfrak{N} \models \neg\phi(\overline{\underset{\sim}{b}}). \end{array}$$

In this case, we will say that ϕ **defines** the set A.

Chaff: It is very important to notice the difference between saying that ϕ represents A and ϕ defines A, which is the same as the difference between $N \vdash$ and $\mathfrak{N} \models$. Notice that any representable set *must* be definable and is defined by any formula that represents it. The converse, however, is not automatic. In fact, the converse is not true. But we're getting ahead of ourselves.

We have mentioned several times that the axiom system N is relatively weak. We will show in this section that N is strong enough to prove some of the $\mathcal{L}_{NT}$ formulas that are true in $\mathfrak{N}$, namely the class of true Σ-sentences. And this will allow us to show that if a set A has a relatively simple *definition*, then the set A will be *representable*.

Recall from Chapter 4 that a formula is a Δ-formula if it contains only bounded quantifiers. Slightly more complicated are the Σ-formulas, which can contain bounded quantifiers and unbounded existential quantifiers, and Π-formulas, which can use both bounded quantifiers and unbounded universal quantifiers.

Example 5.3.9. Here is a perfectly nice example of a Δ-formula ϕ: $(\forall x < t)(x = 0)$. Notice that the denial of ϕ is *not* a Δ-formula, as $\neg(\forall x < t)(x = 0)$ is neither a Σ- nor a Π-formula. But a chain of logical equivalences shows us that $\neg\phi$ is equivalent to a Δ-formula if we just push the negation sign inside the quantifier:

$$\begin{gathered}\neg\phi \\ \neg(\forall x < t)(x = 0) \\ (\exists x < t)\neg(x = 0).\end{gathered}$$

Similarly, we can show that any propositional combination (using $\wedge, \vee, \neg, \rightarrow, \leftrightarrow$) of Δ-formulas is equivalent to a Δ-formula. We will use this fact approximately 215,342 times in the remainder of this book.

A Σ-sentence is, of course, a Σ-formula that is also a sentence. We will be particularly interested in Σ-formulas and Δ-formulas, for we will show if ϕ is a Σ-sentence and $\mathfrak{N} \models \phi$, then the axiom set N is strong enough to provide a deduction of ϕ. Since every Δ-sentence is also a Σ-sentence, any Δ-sentence that is true in $\mathfrak{N}$ is also provable from N.

The first lemma that we will prove shows that N is strong enough to prove that $1 + 1 = 2$. Actually, we already know this since it was proved back in Lemma 2.8.4. We now expand that result and show that if t is any variable-free term, then N proves that t is equal to what it is supposed to be equal to.

Recall that if t is a term, then $t^{\mathfrak{N}}$ is the interpretation of that term in the structure $\mathfrak{N}$. For example, suppose that t is the term $ESSS0SS0$, also known as $SSS0^{SS0}$. Then $t^{\mathfrak{N}}$ would be the number 9, and $\overline{t^{\mathfrak{N}}}$ would be the term $SSSSSSSSS0$. So when this lemma says that N proves $t = \overline{t^{\mathfrak{N}}}$, you should think that N proves $SSS0^{SS0} = SSSSSSSSS0$, which is the same as saying that $N \vdash \overline{3}^{\overline{2}} = \overline{9}$.

Lemma 5.3.10. *For each variable-free term t, $N \vdash t = \overline{t^{\mathfrak{N}}}$.*

Proof. We proceed by induction on the complexity of the term t. If t is the term 0, then $t^{\mathfrak{N}}$ is the natural number 0, and $\overline{t^{\mathfrak{N}}}$ is the term 0. Thus we have to prove that $N \vdash 0 = 0$, which is an immediate consequence of our logical axioms.

If t is $S(u)$, where u is a variable-free term, then the term $\overline{t^{\mathfrak{N}}}$ is identical to the term $S(\overline{u^{\mathfrak{N}}})$. Also, $N \vdash u = \overline{u^{\mathfrak{N}}}$, by the inductive hypothesis, and thus $N \vdash Su = S(\overline{u^{\mathfrak{N}}})$, thanks to the equality axiom (E2). Putting all of this together, we get that $N \vdash t = Su = S(\overline{u^{\mathfrak{N}}}) = \overline{t^{\mathfrak{N}}}$, as needed.

If t is $u + v$, we recall that Lemma 2.8.4 proved that $N \vdash \overline{u^{\mathfrak{N}}} + \overline{v^{\mathfrak{N}}} = \overline{u^{\mathfrak{N}} + v^{\mathfrak{N}}}$. But then $N \vdash t = u + v = \overline{u^{\mathfrak{N}}} + \overline{v^{\mathfrak{N}}} = \overline{u^{\mathfrak{N}} + v^{\mathfrak{N}}} = \overline{t^{\mathfrak{N}}}$, which is what we needed to show. The arguments for terms of the form $u \cdot v$ or u^v are similar, so the proof is complete. □

The next lemma and its corollary will be used in our proof that true Σ-sentences are provable from N.

Lemma 5.3.11 (Rosser's Lemma). *If a is a natural number,*

$$N \vdash (\forall x < \overline{a})\big[\bot \vee x = \overline{0} \vee x = \overline{1} \vee \cdots \vee x = \overline{a-1}\big].$$

Proof. We use induction on a. If $a = 0$, it suffices to prove that $N \vdash \forall x[x < 0 \to \bot]$. By Axiom 9 of N, we know that $N \vdash \neg(x < 0)$, so $N \vdash (x < 0) \to \bot$, as needed.

For the inductive step, suppose that $a = b + 1$. We will be finished if we can show that

$$N \vdash \forall x\big[x < \overline{b+1} \to x = \overline{0} \vee \cdots \vee x = \overline{b}\big].$$

Since $\overline{b+1}$ and $S\overline{b}$ are identical, it suffices to show that

$$N \vdash \forall x\big[x < S\overline{b} \to x = \overline{0} \vee \cdots \vee x = \overline{b}\big].$$

By Axiom 10, we know that $N \vdash x < S\overline{b} \to (x < \overline{b} \vee x = \overline{b})$, and then by the inductive hypothesis, we are finished. □

Corollary 5.3.12. *If a is a natural number, then*

$$N \vdash \big[(\forall x < \overline{a})\phi(x)\big] \leftrightarrow \big[\phi(\overline{0}) \wedge \phi(\overline{1}) \wedge \cdots \wedge \phi(\overline{a-1})\big].$$

Proof. Exercise 11. □

Now we come to the major result of this section, that our axiom system is strong enough to prove all true Σ-sentences.

Proposition 5.3.13. *If $\phi(\underset{\sim}{x})$ is a Σ-formula with free variables $\underset{\sim}{x}$, if $\underset{\sim}{t}$ are variable-free terms, and if $\mathfrak{N} \models \phi(\underset{\sim}{t})$, then $N \vdash \phi(\underset{\sim}{t})$.*

Proof. This is a proof by induction on the complexity of the formula ϕ.

1. If ϕ is atomic, say for example that ϕ is $x < y$ and terms t and u are such that $\mathfrak{N} \models t < u$. Then $t^{\mathfrak{N}} < u^{\mathfrak{N}}$, so by Lemma 2.8.4, $N \vdash \overline{t^{\mathfrak{N}}} < \overline{u^{\mathfrak{N}}}$. But we also know $N \vdash t = \overline{t^{\mathfrak{N}}}$ and $N \vdash u = \overline{u^{\mathfrak{N}}}$, by Lemma 5.3.10, so $N \vdash t < u$, as needed.

2. Negations of atomic formulas are handled in the same manner.

3. If ϕ is $\alpha \vee \beta$ or $\alpha \wedge \beta$, the argument is left to the Exercises.

4. Suppose that $\mathfrak{N} \models \exists x\psi(x)$, where we assume that ψ has only one free variable for simplicity. Then there is a natural number a such that $\mathfrak{N} \models \psi(\overline{a})$, and thus $N \vdash \psi(\overline{a})$ by the inductive hypothesis. But then our second quantifier axiom tells us, as $\overline{a}$ is substitutable for x in ψ, that $N \vdash \exists x\psi$, as needed.

5. Now if $\mathfrak{N} \models (\forall x < u)\psi(x)$, we know by the inductive hypothesis that

$$N \vdash \big[\psi(\overline{0}) \wedge \psi(\overline{1}) \wedge \cdots \wedge \psi(\overline{u^{\mathfrak{N}} - 1})\big].$$

But then by Corollary 5.3.12,

$$N \vdash (\forall x < \overline{u^{\mathfrak{N}}})\psi(x).$$

Thus, since $N \vdash \overline{u^{\mathfrak{N}}} = u$, $N \vdash (\forall x < u)\psi(x)$, as needed.

Thus if $\mathfrak{N} \models \phi(\underset{\sim}{t})$, then $N \vdash \phi(\underset{\sim}{t})$. □

We will say that ϕ is **provable** (from N) if $N \vdash \phi$. And we shall say that ϕ is **refutable** if $N \vdash \neg\phi$.

Suppose that ϕ is a Δ-sentence. If $\mathfrak{N} \models \phi$, since we know that ϕ is also a Σ-sentence, Proposition 5.3.13 shows that $N \vdash \phi$. But suppose that ϕ is false; that is, suppose that $\mathfrak{N} \not\models \phi$. Then $\mathfrak{N} \models \neg\phi$, and $\neg\phi$ is equivalent to a Δ-sentence. Thus by the same argument as above, $N \vdash \neg\phi$. So we have proved the following:

Proposition 5.3.14. *If $\phi(\underset{\sim}{x})$ is a Δ-formula with free variables $\underset{\sim}{x}$, if $\underset{\sim}{t}$ are variable-free terms, and if $\mathfrak{N} \models \phi(\underset{\sim}{t})$, then $N \vdash \phi(\underset{\sim}{t})$. If, on the other hand, $\mathfrak{N} \models \neg\phi(\underset{\sim}{t})$, then $N \vdash \neg\phi(\underset{\sim}{t})$.*

Corollary 5.3.15. *Suppose that $A \subseteq \mathbb{N}^k$ is defined by a Δ-formula $\phi(\underset{\sim}{x})$. Then A is representable.*

Proof. This is immediate from Proposition 5.3.14 and Definition 5.3.1. □

The astute and careful reader will have noticed that Corollary 5.3.15 is an implication and not a biconditional. So the corollary provides us with a useful and convenient way of guaranteeing that a particular set is representable, and we will avail ourselves of that guarantee frequently. The seat-of-the-pants version of the corollary is that a set with a very simple definition is representable. Although we won't be able to prove it until later (Lemma 6.3.3) there is a result that is a nod in the direction of a converse:

Proposition 5.3.16. *Suppose that $A \subseteq \mathbb{N}^k$ is representable. Then there is a Σ-formula that defines A.*

Reading carefully, you are certainly thinking that there must be some representable sets that, although they are Σ-definable, do not have a Δ-definition (if there weren't any, certainly we would have proven that fact in the Corollary above, right?). You are correct, and we will return to this question in the next section. For now, we'll be happy with some practice in defining some sets with Δ-formulas, and thus establishing that those sets are representable. Doing this will keep us busy for much of the rest of this chapter.

Example 5.3.17. Suppose that we look at the even numbers. You might want to define this set by the $\mathcal{L}_{NT}$-formula

$$\phi(x) \text{ is: } (\exists y)(x = y + y).$$

But we can, in fact, do even better than this. We can define the set of evens by a Δ-formula

$$\begin{array}{|l|}\hline \textit{Even}(x) \text{ is:} \\ \qquad\qquad (\exists y \leq x)(x = y + y). \\ \hline \end{array}$$

So now we have a Δ-definition of EVEN, the set of even numbers. (We will try to be consistent and use SMALL CAPITALS when referring to a set of numbers and *Italics* when referring to the $\mathcal{L}_{NT}$-formula that defines that set.) So by Corollary 5.3.15, we see that the set of even numbers is a representable subset of the natural numbers.

Over the next few sections we will be doing a lot of this. We will look at a set of numbers and prove that it is representable by producing a Δ-definition of the set. In many cases, the tricks that we will use to produce the bounds on the quantifiers will be quite impressive.

> *Chaff:* For the rest of this chapter you will see lots of formulas with boxes around them. The idea is that every time we introduce a Δ-definition of a set of numbers, there will be a box around it to set it off.

Example 5.3.18. Take a minute and write $\boxed{\textit{Prime}(x)}$, a Δ-definition of PRIME, the set of prime numbers. Once you have done that, here is a definition of the set of prime pairs, the set of pairs of numbers x and y such that both x and y are prime, and y is the next prime after x:

$$\begin{array}{|l|}\hline \textit{Primepair}(x, y) \text{ is:} \\ \textit{Prime}(x) \wedge \textit{Prime}(y) \wedge (x < y) \wedge \big[(\forall z < y)(\textit{Prime}(z) \rightarrow z \leq x)\big]. \\ \hline \end{array}$$

Notice that *Primepair* has two free variables, as PRIMEPAIR $\subseteq \mathbb{N}^2$, while your formula *Prime* has exactly one free variable. Also notice that all of the quantifiers in each definition are bounded, so we know the definitions are Δ-definitions.

> *Chaff:* We also hope that you noticed that in the definition of *Primepair* we used your formula *Prime*, and we did not try to insert your entire formula every time we needed it— we just wrote *Prime*(x) or *Prime*(y). As you work out the many definitions to follow, it will be essential for you to do the same. Freely

use previously defined formulas and plug them in by using their names. To do otherwise is to doom yourself to unending streams of unintelligible symbols. This stuff gets dense enough as it is. You do not need to make things any harder than they are.

5.3.1 Exercises

1. Show that the set $\{17\}$ is representable by finding a Δ-formula that defines the set. Can you come up with a (probably silly) non-Δ formula that defines the same set?

2. Suppose that $A \subseteq \mathbb{N}$ is representable and represented by the formula $\phi(x)$. Suppose also that $B \subseteq \mathbb{N}$ is representable and represented by $\psi(x)$. Show that the following sets are also representable, and find a formula that represents each:

   (a) $A \cup B$

   (b) $A \cap B$

   (c) The complement of A, $\{x \in \mathbb{N} \mid x \notin A\}$

3. Show that every finite subset of the natural numbers is representable and that every subset of $\mathbb{N}$ whose complement is finite is also representable.

4. Suppose that $f : \mathbb{N}^k \to \mathbb{N}$ is a total function. Show that f is a representable function if and only if f is a representable subset of $\mathbb{N}^{k+1}$.

5. Let $A \subseteq \mathbb{N}$. Define the characteristic function of A, $\chi_A : \mathbb{N} \to \mathbb{N}$ by

$$\chi_A(x) = \begin{cases} 0 & \text{if } x \in A \\ 1 & \text{if } x \notin A \end{cases}$$

   Show that A is a representable subset of $\mathbb{N}$ if and only if χ_A is a representable function.

6. Let $p(x)$ be a polynomial with nonnegative integer coefficients. Show that the set $\{a \in \mathbb{N} \mid p(a) = 0\}$ is representable. After you prove this the obvious way, find a slick way to write the proof. (Or, if you were slick the first time through, find the prosaic way!)

7. Write a Δ-definition for the set DIVIDES. So you must come up with a formula with two free variables, $\boxed{\mathit{Divides}(x, y)}$, which has the property that $\mathfrak{N} \models \mathit{Divides}(\overline{a}, \overline{b})$ if and only if a is a factor of b.

8. Show that the set $\{1, 2, 4, 8, 16, \dots\}$ of powers of 2 is representable.

9. Suppose that you know that $\phi(x, y)$ represents the total function $f : \mathbb{N} \to \mathbb{N}$. Show that $\psi(x, y) :\equiv \phi(x, y) \wedge (\forall z < y)(\neg\phi(x, z))$ also represents f, and furthermore, for each $a \in \mathbb{N}$,

$$N \vdash (\forall y)(\psi(\overline{a}, y) \leftrightarrow y = \overline{f(a)}).$$

[*Suggestion:* After you show that ψ represents f, the second part is equivalent to showing $N \vdash \psi(\overline{a}, \overline{f(a)})$, which is pretty trivial, and then proving that

$$N \vdash \Big[(\phi(\overline{a}, y) \wedge (\forall z < y)(\neg\phi(x, z))) \to y = \overline{f(a)}\Big].$$

So, take as hypotheses N, $\phi(\overline{a}, y)$, and $(\forall z < y)(\neg\phi(x, z))$ and show that there is a deduction of both $\neg\big[\overline{f(a)} < y\big]$ and $\neg\big[y < \overline{f(a)}\big]$. Then the last of the axioms of N will give you what you need. For the details, see [Enderton 72, Theorem 33K].]

10. In the last inductive step of the proof of Lemma 5.3.10, the use of the inductive hypothesis is rather hidden. Please expose the use of the inductive hypothesis and write out that step of the proof more completely. Finish the cases for multiplication and exponentiation.

11. Prove Corollary 5.3.12.

12. Fill in the details of the steps omitted in the inductive proof of Proposition 5.3.13. In the last two cases, how does the argument change if there are more free variables? If, for example, instead of ϕ being of the form $\exists x\psi(x)$, ϕ is of the form $\exists x\psi(x, y)$, does that change the proof?

13. We will say that a formula $\phi(x)$ with one free variable is **positively numeralwise determined** if, for each $a \in \mathbb{N}$, if $\mathfrak{N} \models \phi(a)$ then $N \vdash \phi(\overline{a})$. Say $\phi(x)$ is **numeralwise determined** if both $\phi(x)$ and $\neg\phi(x)$ are positively numeralwise determined. Prove that ϕ represents a set A if and only if ϕ defines A and ϕ is numeralwise determined. To reiterate, a set A is representable if and only if A has a numeralwise determined definition.

14. Show that every atomic formula is numeralwise determined. Then show that the collection of numeralwise determined formulas is closed under $\neg$, $\vee$, $\wedge$ and bounded quantification.

5.4 Representable Functions and Computer Programs

In this section we shall investigate the relationship between representable functions and computer programs. Our discussion will be rather informal and will rely on your intuition about computers and calculations.

One of the reasons that we must be rather informal when discussing computation is that the idea of a calculation is rather vague. In the mid-1930s many mathematicians developed theoretical constructs that tried to capture the idea of a calculable function. Kurt Gödel's recursive functions (now often called computable functions), the Turing Machines of Alan Turing, and Alonzo Church's λ-calculus are three of the best-known models of computability.

One of the reasons that mathematicians accept these formal constructs as accurately modeling the intuitive notion of calculability is that all of the formal analogs of computation that have been proposed have been proved to be equivalent, and each of them is also equivalent to the notion of representability that we defined in the last section. Thus it is known that a function is Turing computable if and only if it is general recursive if and only if it is λ-computable. It is also known that these formal notions are equivalent to the idea of a function being computable on a computer, where we will say that a function f is computable on an idealized computer if there is a computer program P such that if the program P is run with input n, the program will cause the computer to output $f(n)$ and halt.

Thus the situation is like this: On one hand, we have an intuitive idea of what it means for a function to be effectively calculable. On the other hand, we have a slew of formal models of computation, each of which is known to be equivalent to all of the others:

Intuitive Notion	Formal Models
Calculable function	Representable function Computable function λ-Computable function Turing-computable function Computer-computable function $\vdots$

So why do we say that the idea of a calculation is vague? Although all of the *current* definitions are equivalent, for all we know there might be a new definition of calculation that you will come up with tonight over a beer. That new definition will be intuitively correct, in the sense that people who hear your definition agree that it is the "right" definition of what it means for a person to compute something, but your definition may well *not* be equivalent to the current definitions. This will be an earthshaking development and will give logicians and computer scientists plenty to think about for years to come.

You will go down in history as a brilliant person with great insight into the workings of the human mind!

You will win lots of awards and be rich and famous!

All right, we admit it. Not rich. Just famous.

OK. Maybe not famous. But at least well known in logic and computer science circles.

But until you have that beer we will have to go with the current situation, where we have several equivalent definitions that seem to fit our current understanding of the word *computation*. So our idea of what constitutes a computation is imprecise, even though there is precision in the sense that lots of people have thought about what the definition ought to be, and every definition that has been proposed so far has been proved (precisely) to be equivalent to every other definition that has been proposed.

Church's Thesis is simply an expression of the belief that the formal models of computation accurately represent the intuitive idea of a calculable function. We will state the thesis in terms of representability, as we have been working with representable functions and representable sets.

Church's Thesis. *A total function f is calculable if and only if f is representable.*

Now it is important to understand that Church's Thesis is a "thesis" as opposed to a "theorem" and that it will never be a theorem. As an attempt to link an intuitive notion (calculability) and a formal notion (representability) it is not the sort of thing that could *ever* be proved. Proofs require formal definitions, and if we write down a formal definition of calculable function, we will have subverted the meaning of the thesis.

To add another layer to this discussion, consider the function f that assigns to each natural number its natural number square root, if it has one. We will say that $f(n)$ is not defined if n is not a square. So $f(9) = 3$ and $f(10)$ is not defined. This partial function seems to be calculable, and in fact here is some pseudo-code that would compute the output values for f:

```
n <- input
i <- 0
(*) if( i^2 == n){
      output("The square root of ", n, " is ", i)
      halt
      }
i <- i+1
go to (*)
```

To be a little more precise, we will say that a partial function $f : A \subseteq \mathbb{N} \to \mathbb{N}$ is **calculable** if there is an algorithm or computation that, given input $n \in \mathbb{N}$, does exactly one of the following:

- If $f(n)$ is defined, the algorithm computes the correct value of $f(n)$, outputs $f(n)$ and then halts;

- If $f(n)$ is not defined, the algorithm runs forever without halting.

So if the function f is total and calculable, there is an algorithm which will compute $f(n)$ for any input n, but if g is partial and calculable, then g's algorithm will halt when $g(n)$ is defined, but will run forever if $g(n)$ is not defined.

We will say that a set $S \subseteq \mathbb{N}$ is calculable if its characteristic function, χ_S, is calculable. (See Exercise 5 on page 128 for the definition of χ_S.)

Since the study of computation leads rather naturally to the investigation of partial functions, Church's Thesis is often stated in terms of partial functions:

Church's Thesis. *A partial function f is calculable if and only if f is weakly representable.*

The tie between the two versions of Church's Thesis lies in Proposition 5.3.6. Using that proposition, it is easy to see that the total function version of Church's Thesis follows immediately from the partial function version.

For an example of the sort of question that can be addressed by thinking about calculable sets and functions, consider the following:

The class of calculable subsets of $\mathbb{N}$ can be extended by looking at the collection of sets such that there is a computer program which will tell you if a number is an element of the set, but does not have to do anything at all if the number is not an element of the set. Informally, a set $A \subseteq \mathbb{N}$ is said to **semi-calculable** if there is a computer program P such that if $a \in A$, program P returns 0 on input a, and if $a \notin A$, program P does not halt when given input a. You can check that this is equivalent to saying that the partial function $\check{\chi}_A : A \to \mathbb{N}$ that takes on the value 0 for every element of its domain is calculable.

If we accept Church's Thesis, it is easy to argue that set A is representable if and only if both A and $\mathbb{N} - A$ are semi-calculable, as you are asked to do in Exercise 6. Other exercises provide a little more practice in working with semi-calculable sets.

The study of computable functions is an important area of mathematical logic, and emphasizes the tie between logic and computer science. Chapter 7 is devoted to presenting an introduction to computable functions that leads to a proof of Gödel's Incompleteness Theorem. In this setting, the statement of the Incompleteness Theorem amounts to the statement that the collection of sentences that are provable-from-Σ, where Σ is an extension of N that is decidable and true-in-$\mathfrak{N}$, is a semi-computable set that is not computable. Thus there is a significant difference between the collection of computable sets and the collection of semi-computable sets. Another text that emphasizes computability in its treatment of Gödel's Theorem is [Keisler and Robbin 96].

So is Church's Thesis true? We can say that all of the evidence to date seems to suggest that Church's Thesis is true, but we are afraid that is

all the certainty that we can have on that point. We have over 70 years' experience since the statement of the thesis, and over 3000 years since we started computing functions, but that only counts as anecdotal evidence. Even so, most, if not all, of the mathematical community accepts the identification of "computable" with "representable" and thus the community accepts Church's Thesis as an article of faith.

5.4.1 Exercises

1. We defined calculable functions and semi-calculable sets in this section, but the definitions are not set off in their own block and given fancy numbers, like "Definition 5.4.2." Why didn't we make the definitions official-looking like that?

2. Using Church's Thesis, show that $A \subseteq \mathbb{N}$ is calculable if and only if A is representable. Then show that A is semi-calculable if and only if A is weakly representable.

3. (a) Show that $A \subseteq \mathbb{N}$ is semi-calculable if and only if A is listable, where a set is **listable** if there is a computer program L such that L prints out, in some order or another, the elements of A.

   (b) Show $A \subseteq \mathbb{N}$ is calculable if and only if A is listable in increasing order.

4. Suppose that $A \subseteq \mathbb{N}$ is infinite and semi-calculable and show that there is an infinite set $B \subseteq A$ such that B is calculable.

5. Show that $A \subseteq \mathbb{N}$ is semi-calculable if and only if there is a Σ-formula $\phi(x)$ such that ϕ defines A.

6. Use Church's Thesis to show that a set A is representable if and only if both A and $\mathbb{N} - A$ are semi-calculable. [*Suggestion:* First assume that A is representable. This direction is easy. For the other direction, the assumption guarantees the existence of two programs. Think about writing a new program that runs these two programs in tandem—first you run one program for a minute, then you run the second program for a minute. . . .]

5.5 Coding—Naïvely

If you know a child of a certain age, you have undoubtedly run across coded messages of the form

1 14 1 16 16 12 5 1 4 1 25

where letters are coded by numbers associated with their place in the alphabet. If we ignore the spaces above, we can think of the phrase as being

coded by a single number, and that number can have special properties. For example, the code above is not prime and it is divisible by exactly five distinct primes. If we like, we could say that the coding allows us to assert the same statements about the phrase that has been coded. For example, if we take the phrase

The code for this phrase is even

and coded it as a number, you might notice that the code ends in 4, so you might be tempted to say that the phrase was correct in what it asserts.

What we are doing here is representing English statements as numbers, and investigating the properties of the numbers. We can do the same thing with statements of $\mathcal{L}_{NT}$. For example, if we take the sentence

$$= 0S0,$$

we could perhaps code this sentence as the number

$$10424925611375625000000000,$$

and then we can assert things about the number associated with the string. For example, the code for $= 0S0$ is an element of the set of numbers that are divisible by 10, and it is in the set of numbers that are larger than the national debt. What will make all of this interesting is that we can ask if the code for $= 0S0$ is an element of the set of all codes of sentences that can be deduced from N. Then we will be asking if our sentence is a theorem of N. If it is, then we will know that N is inconsistent. If it is not, then N is consistent. Thus we will have reduced the question of consistency of N to a question about numbers and sets!

This is the insight that led Gödel to the Incompleteness Theorem. Given any reasonable set of axioms A, Gödel showed a way to code the phrase

This phrase is not a theorem of A

as a sentence of $\mathcal{L}_{NT}$ and prove that this sentence cannot be in the collection of sentences that are provable from A. So he found a sentence that was true, but not provable. We will, in this chapter and the next, do the same thing.

But first, we will have to establish our coding mechanism. In this section we will not develop our official coding, but rather, a simplified version to give you a taste of the things to come. Let us describe the system we used for the example above.

We started by assigning symbol numbers to the symbols of $\mathcal{L}_{NT}$, as given in Table 5.1. Notice that the symbol numbers are only assigned for the official elements of the language, so if you need to use any abbreviations, such as $\rightarrow$ or $\exists$, you will have to write them out in terms of their definitions.

Then we had to figure out a way to code up sequences of symbols. The idea here is pretty simple, as we will just take the symbol numbers and

Symbol	Symbol Number	Symbol	Symbol Number
$\neg$	1	$+$	13
$\vee$	3	$\cdot$	15
$\forall$	5	E	17
$=$	7	$<$	19
0	9	(	21
S	11	)	23
		v_i	$2i$

Table 5.1: Symbol Numbers for $\mathcal{L}_{NT}$

code them using the scheme that we outlined in Section 4.5. For example, if we look at the expression

$$= 0S0$$

the sequence of symbol numbers this generates is

$$(7, 9, 11, 9),$$

so the code for the sequence would be (remember that we add one to the exponent when we code sequences of numbers)

$$2^8 3^{10} 5^{12} 7^{10},$$

which is also known as

$$10424925611375625000000000,$$

the example that we looked at earlier.

Notice that our coding is effective, in the sense that it is easy, given a number, to find its factorization and thus to find the string that is coded by the number.

> *Chaff:* The word *easy* is used here in its mathematical sense, not in its computer science sense. In reality it can take unbelievably long to factor many numbers, especially numbers of the size that we will discuss.

Now, the problem with all of this is not that you would find it difficult to recognize code numbers, or to decode a given number, or anything like that. Rather, what turns out to be tricky is to show that N, our collection of axioms, is strong enough to be able to *prove* true assertions about the numbers. For example, we would like N to be able to show that the term

$$\overline{10424925611375625000000000}$$

represents a number that is the code for an $\mathcal{L}_{NT}$-sentence. The details of showing that N has this strength will occupy us for the next several sections.

5.5.1 Exercises

1. Decode the message that begins this section.
2. Code up the following $\mathcal{L}_{NT}$-formulas using the method described in this section and in Section 4.5.
   (a) $= +000$
   (b) $= Ev_1Sv_2 \cdot Ev_1v_2v_1$
   (c) $(= 00 \vee (\neg < 00))$
   (d) $(\exists v_2)(< v_2 0)$
3. Find the $\mathcal{L}_{NT}$-formula that is represented by the following numbers. A calculator (or a computer algebra system) will be helpful. Write your answer in a form that normal people can understand—normal people with some familiarity with first-order logic and mathematics, that is.
   (a) 773190132422400000000000000
   (b) 29986008216169640502067200000000
   (c) $2^{22}3^65^77^{24}11^{22}13^817^719^{10}23^{24}$

5.6 Coding Is Representable

A basic part of our coding mechanism will be the ability to code finite sequences of numbers as a single number. A number c is going to be a code for a sequence of numbers $(k_1, k_2, \ldots, k_n)$ if and only if

$$c = \langle k_1, k_2, \ldots, k_n \rangle = 2^{k_1+1}3^{k_2+1} \cdots p_n^{k_n+1},$$

where p_n is the nth prime number.

> *Chaff:* Be careful with the notation here. The sequence of numbers is enclosed by parentheses, while the $\langle \cdot \rangle$ denotes the coding function introduced back in Chapter 4.

We show now that N is strong enough to recognize code numbers. In other words, we want to establish

Proposition 5.6.1. *The collection of numbers that are codes for finite sequences is a representable set.*

Proof. It is easy to write a Δ-definition for the set of code numbers:

$$\begin{array}{l} \textit{Codenumber}(c) \text{ is:} \\ \qquad \textit{Divides}(SS0, c) \wedge (\forall z < c)(\forall y < z) \\ \Big[\big(\textit{Prime}(z) \wedge \textit{Divides}(z, c) \wedge \textit{Primepair}(y, z)\big) \rightarrow \textit{Divides}(y, c)\Big]. \end{array}$$

Notice that *Codenumber*(c) is a formula with one free variable, c. If you look at it carefully, all the formula says is that c is divisible by 2 and if any prime divides c, so do all the earlier primes. Since the definition above is a Δ-definition, Corollary 5.3.15 tells us that the set CODENUMBER is a representable set. □

Since CODENUMBER is representable and *Codenumber* is a Δ-formula, we now know (for example) that

$$N \vdash \mathit{Codenumber}(\overline{18})$$

and

$$N \vdash \neg \mathit{Codenumber}(\overline{45}).$$

Now, suppose that we wanted to take a code number, c, and decode it. To find the third element of the sequence of numbers coded by c, we need to find the exponent of the third prime number. Thus, for N to be able to prove statements about the sequence coded by a number, N will need to be able to recognize the function that takes i and assigns it to the i^{th} prime number, p_i. Proving that this function p is representable is our next major goal.

Proposition 5.6.2. *The function p that enumerates the primes is a representable function.*

Proof. We start by constructing a measure of the primes. A number a will be in the set YARDSTICK if and only if a is of the form $2^1 3^2 5^3 \cdots {p_i}^i$ for some i. So the first few elements of the set are $\{2, 18, 2250, 5402250, \dots\}$.

$$\begin{aligned} &\mathit{Yardstick}(a) \text{ is:} \\ &\qquad \mathit{Codenumber}(a) \wedge \\ &\quad \mathit{Divides}(SS0, a) \wedge \neg \mathit{Divides}(SSS0, a) \wedge \\ &\qquad (\forall y < a)(\forall z < a)(\forall k < a) \\ &\qquad \Big[\big(\mathit{Divides}(z, a) \wedge \mathit{Primepair}(y, z)\big) \rightarrow \\ &\qquad\qquad \big(\mathit{Divides}(y^k, a) \leftrightarrow \mathit{Divides}(z^{Sk}, a)\big)\Big]. \end{aligned}$$

If we unravel this, all we have is that 2 divides a, 4 does not divide a, and if z is a prime number such that z divides a, then the power of z that goes into a is one more than the power of the previous prime that goes into a.

Now it is relatively easy to provide a Δ-definition of the function p:

IthPrime(i, y) is:

$$\begin{aligned} &\mathit{Prime}(y)\wedge \\ &(\exists a \leq (y^i)^i)\big[\mathit{Yardstick}(a) \wedge \mathit{Divides}(y^i, a) \wedge \neg \mathit{Divides}(y^{Si}, a)\big]. \end{aligned}$$

Notice the tricky bound on the quantifier! Here is the thinking behind that bound: If y is, in fact, the i^{th} prime, then here is an $a \in$ YARDSTICK that shows this fact: $a = 2^1 3^2 \cdots y^i$. But then certainly a is less than or equal to $\underbrace{y^i y^i \cdots y^i}_{i \text{ terms}}$, and so $a \leq (y^i)^i$. This bound is, of course, much larger than a (the lone exception being when $i = 1$ and $y = 2$), but we will only be interested in the existence of bounds, and will pay almost no attention to making the bounds precise in any sense. □

> *Chaff:* There is a bit of tension over notation that needs to be mentioned here. Suppose that we wished to discuss the seventeenth prime number, which happens to be 59, and that y is supposed to be equal to 59. The obvious way to assert this would be to state that $y = p_{17}$, but we will tend to use the explicit $\mathcal{L}_{NT}$-formula *IthPrime*$(17, y)$. Our choice will give us a great increase in consistency, as our formulas become rather more complicated over the rest of this chapter. We will tend to write all of our functions in this consistent, if not exactly intuitive manner.

Now we can use the function *IthPrime* to find each element coded by a number:

IthElement(e, i, c) is:

$$\begin{aligned} &\mathit{Codenumber}(c) \wedge (\exists y < c)(\mathit{IthPrime}(i, y)\wedge \\ &\qquad \mathit{Divides}(y^{Se}, c) \wedge \neg \mathit{Divides}(y^{SSe}, c)). \end{aligned}$$

So intuitively, *IthElement*(e, i, c) is true if c is a code and e is the number at position i of the sequence coded by c.

The length of the sequence coded by c is also easily found:

Length(c, l) is:

$$\begin{aligned} &\mathit{Codenumber}(c) \wedge (\exists y < c)\Big[(\mathit{IthPrime}(l, y) \wedge \mathit{Divides}(y, c) \\ &\qquad \wedge (\forall z < c)[\mathit{PrimePair}(y, z) \rightarrow \neg \mathit{Divides}(z, c)])\Big]. \end{aligned}$$

All this says is that if the l^{th} prime divides c and the $(l+1)^{\text{st}}$ prime does not divide c, then the length of the sequence coded by c is l.

5.6.1 Exercise

1. Decide if the following statements are true or false as statements about the natural numbers. Justify your answers.

   (a) $(5, 13) \in$ IthPrime

   (b) $(1200, 3) \in$ Length

   (c) *IthElement*$(\overline{1}, \overline{2}, \overline{3630})$ (Why are there those lines over the numbers?)

5.7 Gödel Numbering

We now change our focus from looking at functions and relations on the natural numbers, where it makes sense to talk about representable sets, to functions mapping strings of $\mathcal{L}_{NT}$-symbols to $\mathbb{N}$. We will establish our coding system for formulas, associating to each $\mathcal{L}_{NT}$-formula ϕ its Gödel number, $\ulcorner \phi \urcorner$. We will make great use of the coding function $\langle \cdot \rangle$ that we defined in Definition 4.5.3.

Definition 5.7.1. The function $\ulcorner \cdot \urcorner$, with domain the collection of finite strings of $\mathcal{L}_{NT}$-symbols and codomain $\mathbb{N}$, is defined as follows:

$$\ulcorner s \urcorner = \begin{cases} \langle 1, \ulcorner \alpha \urcorner \rangle & \text{if } s \text{ is } (\neg\alpha), \text{ where } \alpha \text{ is an } \mathcal{L}_{NT}\text{-formula} \\ \langle 3, \ulcorner \alpha \urcorner, \ulcorner \beta \urcorner \rangle & \text{if } s \text{ is } (\alpha \vee \beta), \text{ where } \alpha \text{ and } \beta \text{ are } \mathcal{L}_{NT}\text{-formulas} \\ \langle 5, \ulcorner v_i \urcorner, \ulcorner \alpha \urcorner \rangle & \text{if } s \text{ is } (\forall v_i)(\alpha), \text{ where } \alpha \text{ is an } \mathcal{L}_{NT}\text{-formula} \\ \langle 7, \ulcorner t_1 \urcorner, \ulcorner t_2 \urcorner \rangle & \text{if } s \text{ is } = t_1 t_2, \text{ where } t_1 \text{ and } t_2 \text{ are terms} \\ \langle 9 \rangle & \text{if } s \text{ is } 0 \\ \langle 11, \ulcorner t \urcorner \rangle & \text{if } s \text{ is } St, \text{ with } t \text{ a term} \\ \langle 13, \ulcorner t_1 \urcorner, \ulcorner t_2 \urcorner \rangle & \text{if } s \text{ is } +t_1 t_2, \text{ where } t_1 \text{ and } t_2 \text{ are terms} \\ \langle 15, \ulcorner t_1 \urcorner, \ulcorner t_2 \urcorner \rangle & \text{if } s \text{ is } \cdot t_1 t_2, \text{ where } t_1 \text{ and } t_2 \text{ are terms} \\ \langle 17, \ulcorner t_1 \urcorner, \ulcorner t_2 \urcorner \rangle & \text{if } s \text{ is } E t_1 t_2, \text{ where } t_1 \text{ and } t_2 \text{ are terms} \\ \langle 19, \ulcorner t_1 \urcorner, \ulcorner t_2 \urcorner \rangle & \text{if } s \text{ is } < t_1 t_2, \text{ where } t_1 \text{ and } t_2 \text{ are terms} \\ \langle 2i \rangle & \text{if } s \text{ is the variable } v_i \\ 3 & \text{otherwise.} \end{cases}$$

Notice that each symbol is associated with its symbol number, as set out in Table 5.1.

Example 5.7.2. Just for practice, let's find $\ulcorner 0 \urcorner$. Just from the chart above, $\ulcorner 0 \urcorner = \langle 9 \rangle = 2^{10} = 1024$. To look at another example, look at $\ulcorner 0 = 0 \urcorner$. Working recursively,

$$\begin{aligned} \ulcorner = 00 \urcorner &= \langle 7, \ulcorner 0 \urcorner, \ulcorner 0 \urcorner \rangle \\ &= \langle 7, 1024, 1024 \rangle \\ &= 2^8 3^{1025} 5^{1025} \end{aligned}$$

Exercise 3 asks you to investigate some of the subtleties of coding as it relates to this last example.

Example 5.7.3. This is a neat function, but the numbers involved get really big, really fast. Suppose that we work out the Gödel number for the formula ϕ, where ϕ is $(\neg = 0S0)$.

Since ϕ is a denial, the definition tells us that

$$\ulcorner \phi \urcorner \text{ is } \langle 1, \ulcorner = 0S0 \urcorner \rangle = 2^2 3^{\ulcorner = 0S0 \urcorner + 1}.$$

So we need to find $\ulcorner = 0S0 \urcorner$, and by the "equals" clause in the definition,

$$\ulcorner = 0S0 \urcorner \text{ is } \langle 7, \ulcorner 0 \urcorner, \ulcorner S0 \urcorner \rangle = 2^8 3^{\ulcorner 0 \urcorner + 1} 5^{\ulcorner S0 \urcorner + 1}.$$

But $\ulcorner 0 \urcorner = 2^{10} = 1024$, and $\ulcorner S0 \urcorner = 2^{12} 3^{\ulcorner 0 \urcorner + 1} = 2^{12} 3^{1025}$. Now we're getting somewhere. Plugging things back in, we get

$$\ulcorner = 0S0 \urcorner \text{ is } 2^8 3^{1025} 5^{[2^{12} 3^{1025} + 1]}$$

so the Gödel number for $(\neg = 0S0)$ is

$$\ulcorner \phi \urcorner \text{ is } 2^2 3^{\left(2^8 3^{1025} 5^{[2^{12} 3^{1025} + 1]} + 1\right)}.$$

Chaff: To get an idea about how large this number is, consider the fact that the exponent on the 5 is $\ulcorner S0 \urcorner = 2^{12} 3^{1025} + 1$, which is

$$\begin{gathered} 458823903732965429493300945942364063611383 5 \\ 337118523487239826617000907251104955407114 16 \\ 244968002327208512018512402196674284003804 68 \\ 284726302476452288447592937167882067262985 94 \\ 576060661169640295861106500088381619676742 48 \\ 714876110453564150536269711030213614452805279 \\ 213722748800276796114884183810302573694405480 \\ 301945785627339339194850085383681785222504546 \\ 327111992210992776215014423059901287305704225 \\ 364360572621118992981982683554087338679406417 0 \\ 563975508362231081323849454313910276632860438529, \end{gathered}$$

approximately 10^{490}.

Now, let us play a bit with the Gödel number of ϕ:

$$\begin{aligned}\ulcorner\phi\urcorner &= 2^2 3^{\left(2^8 3^{1025} 5^{[2^{12} 3^{1025}+1]}+1\right)} \\ &> 3^{5^{[10^{490}]}},\end{aligned}$$

so if we take common logarithms, we see that

$$\log(\ulcorner\phi\urcorner) > 5^{[10^{490}]} \log 3$$

and taking logarithms again,

$$\begin{aligned}\log(\log(\ulcorner\phi\urcorner)) &> 10^{490} \log 5 + \log(\log 3) \\ &> 10^{489}.\end{aligned}$$

Hmm. ... So this means that $\log(\ulcorner\phi\urcorner)$ is bigger than $10^{[10^{489}]}$. But the common logarithm of a number is (approximately) the number of digits that it takes to express the number in base 10 notation, so we have shown that it takes more than $10^{[10^{489}]}$ digits to write out the Gödel number of ϕ. If you remember that a googol is 10^{100} and a googolplex is $10^{10^{100}}$, $\ulcorner\phi\urcorner$ is starting to look like a pretty big number, but it gets better!

To write out a string of $10^{[10^{489}]}$ characters (assuming a million characters per mile, or about 16 characters per inch) would require far more than $10^{[10^{488}]}$ miles, which is far more than $10^{[10^{487}]}$ light years.

Or, to look at it another way, if we assume that we can put about 132 lines of type on an $8\frac{1}{2}$- by 11-inch piece of paper (using both sides), that works out to about $10^{[10^{489}]}$ pieces of paper, and since a ream of paper (500 sheets) is about 2 inches thick, that gives a stack of paper more than $10^{[10^{488}]}$ light years high. Since the age of the universe is currently estimated to be in the tens of billions of years (on the order of 10^{10} years), if we assume that the universe is both Euclidean and spherical, the volume of the universe is less than 10^{40} cubic light years, rather less than the $10^{[10^{488}]}$ cubic light years we would need to store our stack of paper. In short, we don't win any prizes for being incredibly efficient with the coding that we have chosen. What we do win is ease of analysis. The fact that we have chosen to code using a representable function will make our proofs to come much easier to comprehend.

5.7.1 Exercises

1. Evaluate the Gödel number for each of the following:

(a) $(\forall v_3)(v_3 + 0 = v_4)$

(b) $SSSS0$

2. Find the formula or term that is coded by each of the following:

(a) $2^8 3^{\left[2^{14} 3^{\left(2^{18} 3^9 5^{1025}+1\right)} 5^{\left(2^{18} 3^{33} 5^{1025}+1\right)}+1\right]} 5^{\left[2^{18} 3^{129} 5^{1025}+1\right]}$

(b) $2^2 3^{\left(2^{20} 3^{1025} 5^{(2^{12} 3^{1025}+1)}+1\right)}$

(c) $2^6 3^9 5^{\left(2^8 3^9 5^9+1\right)}$

3. Look at the number $c = 2^8 3^{1025} 5^{1025} = \langle 7, \ulcorner 0 \urcorner, \ulcorner 0 \urcorner \rangle$. Find the number e such that $IthElement(e, 2, c)$ is true. Suppose that $d = \langle 3, 1, 4, 5 \rangle = 2^4 3^2 5^5 7^6$. Why is $IthElement(4, 1, d)$ false?

5.8 Gödel Numbers and N

Suppose we asked you if the number

$$\begin{aligned} a \;=\; & 35845617479137924179164136401747192469639 \\ & 33857123846474406114544531958789925746411 \\ & 80793941381251818131650014462814216151784 \\ & 37797240847442423809728350349681982162407 \\ & 86504920777012547391538781481141489453194 \\ & 04814037245506808496961291846992606460711 \\ & 74235438629125412438572089750021624696475 \\ & 32686017988856987542814858951450213588587 \\ & 45976629206001394705081676818056243914838 \\ & 10641798001801554788070758142606590669736 \\ & 02492132671739715266307333432862633253105 \\ & 86590799303228425738618274240361942221 76 \\ & 000000000 \end{aligned}$$

was the Gödel number of an $\mathcal{L}_{NT}$-term. How would you go about finding out? A reasonable approach would be to factor a and try to decode. It turns out that $a = 2^{14} 3^{1025} 5^9$, and since $1024 = 2^{10} = \ulcorner 0 \urcorner$ and $8 = 2^3 = \ulcorner v_1 \urcorner$, you know that a is the Gödel number of the term $+0v_1$. That was easy.

What makes this more interesting is that the above is so easy that N can *prove* that a is the Gödel number of a term. Establishing this fact is the goal of this section.

We will show how to construct certain Δ-formulas, for example the formula $Term(x)$, such that for every natural number a, $\mathfrak{N} \models Term(\overline{a})$ if

and only if a is the Gödel number of a term. Since our formula will be a Δ-formula, this tells us that $N \vdash \mathit{Term}(\overline{a})$ if a is the Gödel number of a term, and $N \vdash \neg \mathit{Term}(\overline{a})$ if a is not the Gödel number of a term.

The problem is going to be in writing down the formula. As you look at the definition of the function $\ulcorner \cdot \urcorner$ in Definition 5.7.1, you can see that the definition is by recursion, and we will need a way to deal with recursive definitions within the constraints of Δ-definitions. We would like to be able to write something like

$$\mathit{Term}(x) \text{ is: } \cdots \vee (\exists y < x)\big[x = 2^{11}3^y \wedge \mathit{Term}(y)\big] \vee \cdots,$$

but this definition is clearly circular. A technical trick will get us past this point.

But we should start at the beginning. You know that the collection of $\mathcal{L}_{NT}$-terms is the closure of the set of variables and the collection of constant symbols under the function symbols. We will begin by showing that the collection of Gödel numbers of variables is representable.

Lemma 5.8.1. *The set*

$$\textsc{Variable} = \{a \in \mathbb{N} \mid a = \ulcorner v \urcorner \textit{ for some variable } v\}$$

is representable.

Proof. It suffices to provide a Δ-definition for VARIABLE:

> $\mathit{Variable}(x)$ is:
> $$(\exists y < x)(\mathit{Even}(y) \wedge 0 < y \wedge x = 2^{Sy}).$$

Notice that we use the fact that if $x = 2^{Sy}$, then $y < x$. It is easy to see that $\mathfrak{N} \models \mathit{Variable}(\overline{a})$ if and only if $a \in$ VARIABLE, so our formula shows that VARIABLE is a representable set. □

To motivate our development of the formula *Term*, consider the term t, where t is $+0Sv_1$. We are used to recognizing that this is a term by looking at it from the outside in: t is a term, as it is the sum of two terms. Now we need to start looking at t from the inside out: t is a term, as there is a sequence of terms, each of which is either a constant symbol, a variable, or constructed from earlier entries in the sequence by application of a function symbol of the appropriate arity. Here is a construction sequence for our term t:

$$(v_1, Sv_1, 0, +0Sv_1).$$

From this construction sequence we can look at the associated sequence of

Gödel numbers:

$$\begin{aligned}(\ulcorner v_1\urcorner, \ulcorner Sv_1\urcorner, \ulcorner 0\urcorner, \ulcorner +0Sv_1\urcorner) &= (\langle 2\rangle, \langle 11, \ulcorner v_1\urcorner\rangle, \langle 9\rangle, \langle 13, \ulcorner 0\urcorner, \ulcorner Sv_1\urcorner\rangle)\\ &= (\langle 2\rangle, \langle 11, 8\rangle, \langle 9\rangle, \langle 13, 1024, \langle 11, \ulcorner v_1\urcorner\rangle\rangle)\\ &= (\langle 2\rangle, \langle 11, 8\rangle, \langle 9\rangle, \langle 13, 1024, \langle 11, 8\rangle\rangle)\\ &= (\langle 2\rangle, \langle 11, 8\rangle, \langle 9\rangle, \langle 13, 1024, 2^{12}3^9\rangle\rangle)\\ &= (8, 2^{12}3^9, 1024, 2^{14}3^{1025}5^{80621569})\\ &= (8, 80621568, 1024, a)\end{aligned}$$

where the large number a is the Gödel number of the term $+0Sv_1$. Now we can code up this sequence of Gödel numbers as a single number

$$c = 2^9 3^{80621569} 5^{1025} 7^{a+1}.$$

Now we can begin to see what our formula $Term(a)$ is going to look like. We will know that a is the Gödel number of a term if there is a number c that is the code for a construction sequence for a term, and the last term in that construction sequence has Gödel number a. To formalize all this, let us begin by defining the collection of construction sequences:

Definition 5.8.2. A finite sequence of $\mathcal{L}_{NT}$-terms $(t_1, t_2, \ldots, t_l)$ is called a **term construction sequence for** t_l if, for each i, $1 \leq i \leq l$, t_i is either a variable, the constant symbol 0, or is one of $t_j, St_j, +t_jt_k, \cdot t_jt_k$, or Et_jt_k, where $j < i$ and $k < i$.

Proposition 5.8.3. *The set*

$$\begin{aligned}&\textsc{TermConstructionSequence} =\\ &\qquad \{(c, a) \mid c \text{ codes a term construction sequence}\\ &\qquad\qquad \text{for the term with Gödel number } a\}\end{aligned}$$

is representable.

Proof. Here is a Δ-definition for the set:

$$\begin{aligned}
&\textit{TermConstructionSequence}(c,a) \text{ is:}\\
&\qquad \textit{CodeNumber}(c)\wedge\\
&(\exists l<c)\Big[\textit{Length}(c,l)\wedge \textit{IthElement}(a,l,c)\wedge\\
&(\forall e<c)(\forall i\le l)\Big(\textit{IthElement}(e,i,c)\rightarrow\\
&\qquad \textit{Variable}(e)\vee e=\overline{2^{\overline{10}}}\vee\\
&\qquad (\exists j<i)(\exists k<i)(\exists e_j<c)(\exists e_k<c)\\
&\qquad \big(\textit{IthElement}(e_j,j,c)\wedge \textit{IthElement}(e_k,k,c)\wedge\\
&\qquad \big[e=e_j\vee e=\overline{2^{\overline{12}}}\cdot\overline{3}^{Se_j}\vee\\
&\qquad e=\overline{2^{\overline{14}}}\cdot\overline{3}^{Se_j}\cdot\overline{5}^{Se_k}\vee\\
&\qquad e=\overline{2^{\overline{16}}}\cdot\overline{3}^{Se_j}\cdot\overline{5}^{Se_k}\vee e=\overline{2^{\overline{18}}}\cdot\overline{3}^{Se_j}\cdot\overline{5}^{Se_k}\big]\big)\Big)\Big].
\end{aligned}$$

This just says that $(c,a)\in$ TermConstructionSequence if and only if c is a code of length l, a is the last number of the sequence coded by c, and if e is an entry at position i of c, then e is either the Gödel number of a variable, the Gödel number of 0, a repeat of an earlier entry, or is the Gödel number that is the result of applying S, $+$, $\cdot$, or E to earlier entries in c. As all of the quantifiers are bounded, this is a Δ-definition, so the set TermConstructionSequence is representable. □

Now it would seem that to define $\textit{Term}(a)$, all we would have to do is to say that a is the Gödel number of a term if there is a number c such that $\textit{TermConstructionSequence}(c,a)$. This is not quite enough, as the quantifier $\exists c$ is not bounded. In order to write down a Δ-definition of *Term*, we will have to get a handle on how large codes for construction sequences have to be.

Lemma 5.8.4. *If t is an $\mathcal{L}_{NT}$-term and $\ulcorner t\urcorner = a$, then the number of symbols in t is less than a.*

Proof. The proof is by induction on the complexity of t. Just to give you an idea of how true the lemma is, consider the example of t, where t is $S0$. Then t has two symbols, while $\ulcorner t\urcorner = a = 2^{12}3^{1025}$, which is just a little bigger than 2. □

Lemma 5.8.5. *If t is a term, the length of the shortest construction sequence of t is less than or equal to the number of symbols in t.*

Proof. Again, use induction on the complexity of t. □

These lemmas tell us that if a is the Gödel number of a term, then there is a construction sequence of that term whose length is less than a.

Lemma 5.8.6. *Suppose that t is an $\mathcal{L}_{NT}$-term, u is a subterm of t. (In other words, u is a substring of t, u is also an $\mathcal{L}_{NT}$-term, and u is not identical to t.) Then $\ulcorner u \urcorner < \ulcorner t \urcorner$.*

Proof. Exercise 4. □

Lemma 5.8.7. *If a is a natural number greater than or equal to 1, then $p_a \leq 2^{a^a}$, where p_a is the ath prime number.*

Proof. Exercise 5. □

Now we have enough to give us our bound on the code for the shortest construction sequence for a term t with Gödel number $\ulcorner t \urcorner = a$. Any such construction sequence must look like

$$(t_1, t_2, \ldots, t_k = t),$$

where $k \leq a$ and each t_i is a subterm of t. But then the code for this construction sequence is

$$\begin{aligned}
c &= \langle \ulcorner t_1 \urcorner, \ulcorner t_2 \urcorner, \ldots, \ulcorner t \urcorner \rangle \\
&= 2^{\ulcorner t_1 \urcorner + 1} 3^{\ulcorner t_2 \urcorner + 1} \cdots p_k^{\ulcorner t \urcorner + 1} \\
&\leq 2^{a+1} 3^{a+1} \cdots p_k^{a+1} \\
&\leq \underbrace{p_k^{a+1} p_k^{a+1} \cdots p_k^{a+1}}_{k \text{ terms}} \\
&\leq \underbrace{p_a^{a+1} p_a^{a+1} \cdots p_a^{a+1}}_{a \text{ terms}} \\
&= \left[\left(2^{a^a} \right)^{a+1} \right]^a \\
&= \left(2^{a^a} \right)^{a^2 + a},
\end{aligned}$$

which gives us our needed bound.

We are finally at a position where we can give a Δ-definition of the collection of Gödel numbers of $\mathcal{L}_{NT}$-terms:

> *Term*(a) is:
>
> $$\left(\exists c < \left(\overline{2}^{a^a} \right)^{a^{\overline{2}} + a} \right) \mathit{TermConstructionSequence}(c, a).$$

So the set

$$\textsc{Term} = \{a \in \mathbb{N} \mid a \text{ is the Gödel number of an } \mathcal{L}_{NT}\text{-term}\}$$

is a representable set, and thus N has the strength to prove, for any number a, either *Term*$(\overline{a})$ or $\neg$*Term*$(\overline{a})$. In Exercise 6 we will ask you to show that the set Formula is also representable.

5.8.1 Exercises

1. Assume that ϕ is a formula of $\mathcal{L}_{NT}$. Which of the following are also $\mathcal{L}_{NT}$-formulas? For the ones that are not formulas, why are they not formulas?

   - $Term(\phi)$
   - $Term(\ulcorner \phi \urcorner)$
   - $Term\left(\overline{\ulcorner \phi \urcorner}\right)$

2. Suppose that in the definition of *TermConstructionSequence*, you saw the following string:

   $$\cdots \vee e = \overline{2^{11} \cdot 3^{e_j}} \vee \cdots$$

   Would that be a part of a legal $\mathcal{L}_{NT}$-formula? How do you know? What if the string were

   $$\cdots \vee e = \overline{2}^{\overline{11}} \cdot \overline{3}^{\overline{e_j}} \vee \cdots ?$$

3. Prove Lemma 5.8.4.

4. Prove Lemma 5.8.6.

5. Prove Lemma 5.8.7 by induction. For the inductive step, if you are trying to prove that $p_{n+1} \leq 2^{(n+1)^{n+1}}$, use the fact that p_{n+1} is less than or equal to the smallest prime factor of $(p_1 p_2 \cdots p_n) - 1$.

6. A proof similar to the proof that TERM is representable will show that

   $$\text{FORMULA} = \{a \in \mathbb{N} \mid a \text{ is the Gödel number of an } \mathcal{L}_{NT}\text{-formula}\}$$

   is also representable. Carefully supply the needed details and define the formula $\boxed{Formula(f)}$. You will probably have to define the formula $FormulaConstructionSequence(c, f)$ and estimate the length of such sequences as part of your exposition.

5.9 NUM and SUB Are Representable

In our proof of the Self-Reference Lemma in Section 6.2, we will have to be able to substitute the Gödel number of a formula into a formula. To do this it will be necessary to know that a couple of functions are representable, and in this section we outline how to construct Δ-definitions of those functions. First we work with the function Num.

Recall that $\overline{a}$ is the numeral representing the number a. Thus, $\overline{2}$ is $SS0$. Since $SS0$ is an $\mathcal{L}_{NT}$-term, it has a Gödel number, in this case

$$\ulcorner SS0 \urcorner = \langle 11, \ulcorner S0 \urcorner \rangle = \langle 11, 2^{12}3^{1025} \rangle = 2^{12}3^{2^{12}3^{1025}+1}.$$

The function Num that we seek will map 2 to the Gödel number of its $\mathcal{L}_{NT}$-numeral, $2^{12}3^{2^{12}3^{1025}+1}$. So $\mathrm{Num}(a) = \ulcorner \overline{a} \urcorner$.

To write a Δ-definition $Num(a, y)$ we will start, once again, with a construction sequence, but this time we will construct the numeral $\overline{a}$ using a particular term construction sequence. To give an example, consider the case $a = 2$. We will use the easiest construction sequence that we can think of, namely

$$(0, S0, SS0),$$

which gives rise to the sequence of Gödel numbers

$$(\ulcorner 0 \urcorner, \ulcorner S0 \urcorner, \ulcorner SS0 \urcorner) = (1024, 2^{12}3^{1025}, 2^{12}3^{2^{12}3^{1025}+1}),$$

which is coded by the number

$$c = 2^{[1025]}3^{[2^{12}3^{1025}+1]}5^{[2^{12}3^{2^{12}3^{1025}+1}+1]}.$$

Notice that the length of the construction sequence here is 3, and in general the construction sequence will have length $a + 1$ if we seek to code the construction of the Gödel number of the numeral associated with the number a. (That is a very long sentence, but it does make sense if you work through it carefully.)

You are asked in Exercise 1 to write down the formula

$$\boxed{NumConstructionSequence(c, a, y)}$$

as a Δ-formula. The idea is that

$$\mathfrak{N} \models NumConstructionSequence(c, a, y)$$

if and only if c is the code for a construction sequence of length $a + 1$ with last element $y = \ulcorner \overline{a} \urcorner$.

Now we would like to define the formula $Num(a, y)$ in such a way that $Num(a, y)$ is true if and only if y is $\ulcorner \overline{a} \urcorner$, and as in Section 5.8, the formula $(\exists c)NumConstructionSequence(c, a, y)$ does not work, as the quantifier is unbounded. So we must find a bound for c.

If $(c, a, y) \in$ NumConstructionSequence, then we know that c codes a construction sequence of length $a + 1$ and c is of the form

$$c = 2^{\ulcorner t_1 \urcorner + 1}3^{\ulcorner t_2 \urcorner + 1} \cdots p_{a+1}^{y+1},$$

where each t_i is a subterm of $\overline{a}$. By Lemma 5.8.6 and Lemma 5.8.7, we know that $\ulcorner t_i \urcorner \leq y$ and $p_{a+1} \leq 2^{(a+1)^{(a+1)}}$, so

$$c \leq \underbrace{2^{y+1}3^{y+1} \ldots p_{a+1}^{y+1}}_{a+1 \text{ terms}} \leq (p_{a+1})^{(a+1)(y+1)} \leq \left(2^{(a+1)^{(a+1)}}\right)^{(a+1)(y+1)}.$$

This gives us our needed bound on c. Now we can define

$Num(a, y)$ is:

$$\left(\exists c < \left(\overline{2}^{(a+\overline{1})^{(a+\overline{1})}}\right)^{(a+\overline{1})\cdot(y+\overline{1})}\right) NumConstructionSequence(c, a, y).$$

The next formulas that we need to discuss will deal with substitution. We will define two Δ-formulas:

1. $TermSub(u, x, t, y)$ will represent substitution of a term for a variable in a term (u_t^x).
2. $Sub(f, x, t, y)$ will represent substitution of a term for a variable in a formula (ϕ_t^x).

Specifically, we will show that $\mathfrak{N} \models TermSub(\ulcorner u \urcorner, \ulcorner x \urcorner, \ulcorner t \urcorner, y)$ if and only if y is the Gödel number of u_t^x, where it is assumed that u and t are terms and x is a variable. Similarly, $Sub(\ulcorner \phi \urcorner, \ulcorner x \urcorner, \ulcorner t \urcorner, y)$ will be true if and only if ϕ is a formula and $y = \ulcorner \phi_t^x \urcorner$.

We will develop *TermSub* carefully and outline the construction of *Sub*, leaving the details to the reader.

First, let us look at an example. Suppose that u is the term $+ \cdot 0S0x$. Then a construction sequence for u could look like

$$(0, x, S0, \cdot 0S0, + \cdot 0S0x).$$

If t is the term $SS0$, then u_t^x is $+ \cdot 0S0SS0$, and we will find a construction sequence for this term in stages.

The first thing we will do is look at the sequence (no longer a construction sequence) that we obtain by replacing all of the x's in u's construction sequence with t's:

$$(0, SS0, S0, \cdot 0S0, + \cdot 0S0SS0).$$

This fails to be a construction sequence, as the second term of the sequence is illegal. However, if we precede this whole thing with the construction sequence for t, all will be well (recall that we are allowed to repeat elements in a construction sequence):

$$(0, S0, SS0, 0, SS0, S0, \cdot 0S0, + \cdot 0S0SS0).$$

So to get all of this to work, we have to do three things:

1. We must show how to change u's construction sequence by replacing the occurrences of x with t.

2. We must show how to put one construction sequence in front of another.

3. We must make sure that all of our quantifiers are bounded throughout, so that our final formula *TermSub* is a Δ-formula.

So, first we define a formula $\mathit{TermReplace}(c, u, d, x, t)$ such that if c is the code for a construction sequence of a term with Gödel number u, then d is the code of the sequence (probably not a construction sequence) that results from replacing each occurrence of the variable with Gödel number x by the term with Gödel number t. In the following definition, the idea is that e_i and a_i are the entries at position i of the sequence coded by c and d, respectively, and if we look at it line by line we see: c codes a construction sequence for the term coded by u, d codes a sequence; the lengths of the two sequences are the same; if e_i and a_i are the i^{th} entries in sequence c and d, respectively, then e_i is x if and only if a_i is t; e_i is another variable if and only if a_i is the same variable; e_i codes 0 if and only if a_i also codes 0; e_i codes the successor of a previous e_j if and only if a_i codes the successor of the corresponding a_j; and so on.

$\mathit{TermReplace}(c, u, d, x, t)$ is:

$$\mathit{TermConstructionSequence}(c, u) \wedge \mathit{Codenumber}(d) \wedge$$
$$(\exists l < c)\Big[\mathit{Length}(c, l) \wedge \mathit{Length}(d, l) \wedge$$
$$(\forall i \leq l)(\forall e_i < c)(\forall a_i < d)$$
$$\Big((\mathit{IthElement}(e_i, i, c) \wedge \mathit{IthElement}(a_i, i, d)) \rightarrow$$
$$\big[([(\mathit{Variable}(e_i) \wedge e_i = x) \leftrightarrow a_i = t] \wedge$$
$$(\mathit{Variable}(e_i) \wedge e_i \neq x) \leftrightarrow (\mathit{Variable}(a_i) \wedge a_i = e_i)) \wedge$$
$$(e_i = \overline{2^{\overline{10}}} \leftrightarrow a_i = \overline{2^{\overline{10}}}) \wedge$$
$$(\forall j < i)\Big[\big((\exists e_j < c)\mathit{IthElement}(e_j, j, c) \wedge e_i = \overline{2^{\overline{12}}} \cdot \overline{3}^{e_j}\big) \rightarrow$$
$$\big((\exists a_j < d)\mathit{IthElement}(a_j, j, d) \wedge a_i = \overline{2^{\overline{12}}} \cdot \overline{3}^{a_j}\big)\Big] \wedge$$
$$\vdots$$
$$\text{(Similar clauses for } +, \cdot, \text{ and } E.)$$
$$\vdots$$
$$\big]\Big)\Big].$$

Now that we know how to destroy a construction sequence for u by replacing all occurrences of x with t, we have to be able to put a term construction sequence for t in front of our sequence to make it a construction sequence again. So suppose that d is the code for the sequence obtained by replacing x by t, and that b is the code for the term construction sequence for t. Essentially, we want a to code d appended to b. So the length of a is the sum of the lengths of b and d, and the first l elements of the a sequence should be the same as the b sequence, where the length of the b sequence is l, and the rest of the a sequence should match, entry by entry, the d sequence:

$$
\begin{aligned}
&\mathit{Append}(b,d,a) \text{ is:}\\
&\quad \mathit{Codenumber}(b) \wedge \mathit{Codenumber}(d) \wedge \mathit{Codenumber}(a) \wedge\\
&\quad (\exists l < b)(\exists m < d)\Big(\mathit{Length}(l,b) \wedge\\
&\quad \mathit{Length}(m,d) \wedge \mathit{Length}(l+m,a) \wedge\\
&\quad (\forall e < a)(\forall i \le l)\big[\mathit{IthElement}(e,i,a) \leftrightarrow \mathit{IthElement}(e,i,b)\big] \wedge\\
&\quad (\forall e < a)(\forall j \le m)\big[0 < j \rightarrow\\
&\quad (\mathit{IthElement}(e,l+j,a) \leftrightarrow \mathit{IthElement}(e,j,d))\big]\Big).
\end{aligned}
$$

Now we are ready to define the formula $\mathit{TermSub}(u,x,t,y)$ that is supposed to be true if and only if y is the (Gödel number of the) term that results when you take the term (with Gödel number) u and replace (the variable with Gödel number) x by (the term with Gödel number) t. A first attempt at a definition might be

$$
\begin{aligned}
(\exists a)(\exists b)(\exists d)(\exists c)\big[&\mathit{TermConstructionSequence}(c,u) \wedge\\
&\mathit{TermConstructionSequence}(b,t) \wedge\\
&\mathit{TermReplace}(c,u,d,x,t) \wedge \mathit{Append}(b,d,a) \wedge\\
&\mathit{TermConstructionSequence}(a,y)\big].
\end{aligned}
$$

This is nice, but we need to bound all of the quantifiers. Bounding b and d is easy: They are smaller than a. As for c, we showed when we defined the formula $\mathit{Term}(a)$ on page 146 that the term coded by u must have a construction sequence coded by a number less than $\left(2^{u^u}\right)^{u^2+u}$. So all that is left is to find a bound on a. Notice that we can assume that b codes a sequence of length less than or equal to t, the last entry of b, and similarly, c codes a sequence of length less than or equal to u. Since the sequence coded by d has the same length as the sequence coded by c, and as a codes b's sequence followed by d's, the length of the sequence coded by a is less than or equal to $u+t$. So

$$a \le 2^{e_1}3^{e_2}\cdots p_{t+u}^{y}.$$

Now we mirror the definition of *TermConstructionSequence*. As each entry of a can be assumed to be less than or equal to y, we get

$$\begin{aligned} a &\leq 2^y 3^y \dots p_{t+u}^y \\ &\leq [(p_{t+u})^y]^{t+u} \\ &\leq \left(\left[2^{t+u^{t+u}}\right]^y\right)^{t+u}. \end{aligned}$$

So our Δ-definition of TermSub is

$$\begin{aligned} &\textit{TermSub}(u,x,t,y) \text{ is:} \\ &\left(\exists a < \left(\left[\overline{2}^{t+u^{t+u}}\right]^y\right)^{t+u}\right)(\exists b < a)(\exists d < a)\left(\exists c < \left(\overline{2}^{u^u}\right)^{u^{\overline{2}}+u}\right) \\ &\qquad \big[\textit{TermConstructionSequence}(c,u) \wedge \\ &\qquad\quad \textit{TermConstructionSequence}(b,t) \wedge \\ &\qquad \textit{TermReplace}(c,u,d,x,t) \wedge \textit{Append}(b,d,a) \wedge \\ &\qquad\qquad \textit{TermConstructionSequence}(a,y)\big]. \end{aligned}$$

Chaff: Garbage cases. What an annoyance. Our claim in the paragraph preceding the definition of *TermSub* that each entry of a will be less than or equal to y might not be correct if $y = u_t^x = u$, so the substitution is vacuous. The reason for this is that the entries of a would include a construction sequence for t, which might be huge, while y might be relatively small. For example, we might have $u = 0$ and $t = v_{123456789}$ and $x = v_1$. In Exercise 3 you are invited to figure out the slight addition to the definition of *TermSub* that takes care of this.

Now we outline the construction of the formula $\textit{Sub}(f,x,t,y)$, which is to be true if f is the Gödel number of a formula ϕ and y is $\ulcorner \phi_t^x \urcorner$. The idea is to take a formula construction sequence for ϕ and follow it by a copy of the same construction sequence where we systematically replace the occurrences of x with t's. You do have to be a little careful in your replacements, though. If you compare Definitions 1.8.1 and 1.8.2, you can see that the rules for replacing variables by terms are a little more complicated in the formula case than in the term case, particularly when you are substituting in a quantified formula. But the difficulties are not too bad.

So if b codes the construction sequence for ϕ, if d codes the sequence that you get after replacing the x's by t's, and if $\textit{Append}(b,d,a)$ holds, then a will code up a construction sequence for ϕ_t^x. After you deal with the bounds, you will have a Δ-formula along the lines of

$Sub(f, x, t, y)$ is:
$$(\exists a < \text{Bound})(\exists b < a)(\exists d < a)$$
$$FormulaConstructionSequence(b, f) \wedge$$
$$FormulaReplace(b, f, d, x, t) \wedge Append(b, d, a) \wedge$$
$$FormulaConstructionSequence(a, y).$$

5.9.1 Exercises

1. Write the formula $NumConstructionSequence(c, a, y)$. Make sure that your formula is a Δ-formula with the variables c, a, and y free. [*Suggestion:* You might want to model your answer on the construction of *TermConstructionSequence*.]

2. Write out the "Similar clauses" in the definition of *TermReplace*.

3. Change the definition of *TermSub* to take care of the problem mentioned on page 152. [*Suggestion:* The problem occurs only if the substitution is vacuous. So there are two cases. Either u and y are different, in which case our definition is fine, or u and y are equal. What do you need to do then? So we suggest that your answer should be a disjunction, something like

$MyTermSub(u, x, t, y)$ is:
$$\big[(u \neq y) \wedge TermSub(u, x, t, y)\big] \vee$$
$$\big[(u = y) \wedge (\text{Something Brilliant})\big].$$

4. Write out the details of the formula *Sub*.

5.10 Definitions by Recursion Are Representable

If you look at the definition of a term (Definition 1.3.1), the definition of formula (Definition 1.3.3), the definition of u_t^x (Definition 1.8.1), and the definition of ϕ_t^x (Definition 1.8.2), you will notice that all of these definitions were definitions "by recursion." For example, in the definition of a term, you see the phrase

> ... t is $ft_1t_2 \ldots t_n$, where f in an n-ary function symbol of $\mathcal{L}$ and each of the t_i are terms of $\mathcal{L}$

so a term can have constituent parts that are themselves terms. In the last two sections we have used the device of construction sequences to show

that the sets TERM, FORMULA, TERMSUB, and SUB are representable sets. In this section we outline a proof that all such sets of strings, defined "by recursion," give rise to sets of Gödel numbers that are representable. It will be clear from our exposition that a more general statement of our theorem could be proved, but what we present will be sufficient for our needs.

Definition 5.10.1. A string of symbols s from a first-order language $\mathcal{L}$ is called an **expression** if s is either a term of $\mathcal{L}$ or a formula of $\mathcal{L}$.

Theorem 5.10.2. *Suppose that we have a set of $\mathcal{L}_{NT}$-expressions, which we will call Set, defined as follows: An expression s is an element of Set if and only if:*

1. *s is an element of BaseCaseSet, or*
2. *There is an expression t, a proper substring of s, such that (t, s) is an element of ConstructionSet.*

If the sets of strings BaseCaseSet and ConstructionSet give rise to sets of Gödel numbers BASECASESET *and* CONSTRUCTIONSET *that are defined by Δ-formulas, then the set*

$$\text{SET} = \{\ulcorner s \urcorner \mid s \in Set\}$$

is representable, and has a Δ-definition Set.

Chaff: Try to keep the various typefaces straight:

- *Set* is a bunch of $\mathcal{L}_{NT}$-expressions—strings of symbols from $\mathcal{L}_{NT}$.
- SET is a set of natural numbers—the Gödel numbers of the strings in *Set*.
- *Set* is an $\mathcal{L}_{NT}$-formula such that $\mathfrak{N} \models Set(\overline{a})$ if and only if there is an $s \in Set$ such that $\ulcorner s \urcorner = a$.

Proof. We follow very closely the proof of the representability of the set TERM, which begins on page 143. As you worked through Exercise 6 in Section 5.8.1, you saw that you could prove the analogs of Lemmas 5.8.4 through 5.8.6 for formulas, so you have established the following lemma:

Lemma 5.10.3. *Suppose that s is an $\mathcal{L}_{NT}$-expression and u is a substring of s that is also an expression. Then*

1. *If $\ulcorner s \urcorner = a$, then the number of symbols in s is less than or equal to a.*

2. *The length of the shortest construction sequence of s is less than or equal to the number of symbols in s.*

3. $\ulcorner u \urcorner < \ulcorner s \urcorner$.

Now we can write a Δ-definition of SETCONSTRUCTIONSEQUENCE and then use this lemma to show SET is representable:

$$\begin{aligned} &\textit{SetConstructionSequence}(c,a) \text{ is:} \\ &\qquad \textit{CodeNumber}(c) \wedge \\ &\qquad (\exists l < c)\Big[\textit{Length}(c,l) \wedge \textit{IthElement}(a,l,c) \wedge \\ &\qquad\quad (\forall i \le l)(\forall e < c)\Big(\textit{IthElement}(e,i,c) \rightarrow \\ &\qquad\qquad \textit{BaseCaseSet}(e) \vee \\ &\qquad\qquad (\exists j < i)(\exists e_j < c) \\ &\qquad\qquad \big(\textit{IthElement}(e_j,j,c) \wedge \textit{ConstructionSet}(e_j,e)\big)\Big)\Big]. \end{aligned}$$

We know, by assumption, that there are Δ-formulas *BaseCaseSet* and *ConstructionSet* that define the representable sets BASECASESET and CONSTRUCTIONSET. Thus, all of the quantifiers in the definition above are bounded, and *SetConstructionSequence* is a Δ-formula.

As before we can use Lemmas 5.8.5 and 5.8.7 to bound the size of the shortest construction sequence for a: By the same argument as on page 146, there is a construction sequence coded by a number c such that $c < \left(2^{a^a}\right)^{a^2+a}$. So we define

$$\begin{aligned} &\textit{Set}(a) \text{ is:} \\ &\qquad \left(\exists c < \left(\overline{2}^{a^a}\right)^{a^{\overline{2}}+a}\right) \textit{SetConstructionSequence}(c,a) \end{aligned}$$

and we have a Δ-definition of the set SET, finishing our proof. $\square$

This is a wonderful theorem, as it saves us lots of work. Merely by noting that their definitions fit the requirements of Theorem 5.10.2, the following sets all turn out to be representable and have Δ-definitions:

1. FREE, where $(x,f) \in$ FREE if and only if x is the Gödel number of a variable that is free in the formula with Gödel number f.

2. SUBSTITUTABLE, where $(t,x,f) \in$ SUBSTITUTABLE if and only if t is the Gödel number of a term that is substitutable for the variable with Gödel number x in the formula with Gödel number f.

5.10.1 Exercises

1. Work through the details (including the needed modification of Theorem 5.10.2) and show that both FREE and SUBSTITUTABLE are representable.

2. Suppose that the function $f : \mathbb{N}^{k+1} \to \mathbb{N}$ is defined as follows:

$$f(\underset{\sim}{a}, 0) = g(\underset{\sim}{a})$$
$$f(\underset{\sim}{a}, b+1) = h(\underset{\sim}{a}, b, f(\underset{\sim}{a}, b)),$$

   where g and h are representable functions represented by Δ-formulas. Show that f is a representable function. [*Suggestion:* One approach is to define the formula $f\mathit{ConstructionSequence}(c, \underset{\sim}{a}, l)$, with the idea that the sequence coded by c will be $(f(0), f(1), \ldots f(l))$. Or, you might try to fit this situation into a theorem along the lines of Theorem 5.10.2.]

3. Use Exercise 2 to show that the following functions are representable:

   (a) The factorial function $n!$

   (b) The Fibonacci function F, where $F(1) = F(2) = 1$, and for $k \geq 3$, $F(k) = F(k-1) + F(k-2)$

   (c) The function $a \uparrow i$, where $a \uparrow 0 = 1$ and $a \uparrow (j+1) = a^{a \uparrow j}$ (You should also compute a few values, along the lines of $2 \uparrow 3$, $2 \uparrow 4$, and $2 \uparrow 5$.)

5.11 The Collection of Axioms Is Representable

In this section we will exhibit two Δ-formulas that are designed to pick out the axioms of our deductive system.

Proposition 5.11.1. *The collection of Gödel numbers of the axioms of N is representable.*

Proof. The formula *AxiomOfN* is easy to describe. As there are only a finite number of N-axioms, a natural number a is in the set AXIOMOFN if and only if it is one of a finite number of Gödel numbers. Thus

AxiomOfN(a) is:

$$a = \overline{\ulcorner (\forall x) \neg Sx = 0 \urcorner} \ \vee$$
$$a = \overline{\ulcorner (\forall x)(\forall y)\big[Sx = Sy \to x = y\big] \urcorner} \ \vee$$
$$\vdots$$
$$\vee\, a = \overline{\ulcorner (\forall x)(\forall y)\big[(x < y) \vee (x = y) \vee (y < x)\big] \urcorner}.$$

(To be more-than-usually picky, we need to change the x's and y's to v_1's and v_2's, but you can do that.) □

Proposition 5.11.2. *The collection of Gödel numbers of the logical axioms is representable.*

Proof. The formula that recognizes the logical axioms is more complicated than the formula *AxiomOfN* for two reasons. The first is that there are infinitely many logical axioms, so we cannot just list them all. The second reason that this group of axioms is more complicated is that the quantifier axioms depend on the notion of substitutability, so we will have to use our results from Section 5.10.

Quite probably you are at a point where you could turn to Section 2.3.3 and write down a Δ-definition of the set LogicalAxiom. To do so would be a worthwhile exercise. But if you are feeling lazy, here is an attempt:

LogicalAxiom(a) is:

$$(\exists x < a)(\mathit{Variable}(x) \wedge a = \overline{2}^{\overline{8}}\overline{3}^{Sx}\overline{5}^{Sx}) \vee$$

$$(\exists x, y < a)\Bigg(\mathit{Variable}(x) \wedge \mathit{Variable}(y) \wedge$$

$$a = \overline{2}^{\overline{4}}\overline{3}^{\left(\overline{2}^{\overline{2}}\overline{3}^{(2^{\overline{8}}\overline{3}^{Sx}\overline{5}^{Sy}+1)}+1\right)}\overline{5}^{\left(\overline{2}^{\overline{8}}\overline{3}^{\overline{2}^{\overline{12}}\overline{3}^{Sx}+1}\overline{5}^{\overline{2}^{\overline{12}}\overline{3}^{Sy}+1}+1\right)}\Bigg) \vee$$

$$\Bigg((\exists x_1, x_2, y_1, y_2 < a)(\mathit{Variable}(x_1) \wedge \cdots \wedge \mathit{Variable}(y_2) \wedge$$

$$a = \text{Ugly Mess saying}\big[(x_1 = y_1) \wedge (x_2 = y_2)\big] \rightarrow$$

$$(x_1 + x_2 = y_1 + y_2))\Bigg) \vee$$

$$\vdots$$

(Similar clauses coding up (E2) and (E3) for $\cdot$, E, $=$, and $<$)

$$\vdots$$

$$\vee\, (\exists f, x, t, y < a)\Bigg(\mathit{Formula}(f) \wedge \mathit{Variable}(x) \wedge \mathit{Term}(t) \wedge$$

$$\mathit{Substitutable}(t, x, f) \wedge \mathit{Sub}(f, x, t, y) \wedge$$

$$a = \overline{2}^{\overline{4}}\overline{3}^{\left(\overline{2}^{\overline{2}}\overline{3}^{(2^{\overline{6}}\overline{3}^{Sx}\overline{5}^{Sf}+1)}+1\right)}\overline{5}^{Sy}\Bigg) \vee$$

(Similar clause for Axiom (Q2)).

To look at this in a little more detail, the first clause of the formula is

supposed to correspond to axiom (E1): $x = x$ for each variable x. So a is the Gödel number of an axiom of this form if there is some x that is the *Gödel number of* a variable [which is what $Variable(x)$ says] such that a is the *Gödel number of* a formula that looks like

$$\text{variable with Gödel number } x = \text{variable with Gödel number } x.$$

But the Gödel number for this formula is just $2^8 3^{Sx} 5^{Sx}$, so that is what we demand that a equal.

The second clause covers axioms of the form (E2), when the function f is the function S. We demand that a be the code for the formula

$$(v_i = v_j) \to Sv_i = Sv_j,$$

where $x = \ulcorner v_i \urcorner$ and $y = \ulcorner v_j \urcorner$. After you fuss with the coding, you come out with the expression shown. The other clauses of type (E2) and (E3) are similar.

The last clause that is written out is for the quantifier axiom (Q1). After demanding that the term coded by t be substitutable for the variable coded by x in the formula coded by f and that y be the code for the result of substituting in that way, the equation for a is nothing more than the analog of $(\forall x\phi) \to \phi_t^x$. □

5.11.1 Exercise

1. Complete the definition of the formula *LogicalAxiom*.

5.12 Coding Deductions

It is probably difficult to remember at this point of our journey, but our goal is to prove the Incompleteness Theorem, and to do that we need to write down an $\mathcal{L}_{NT}$-sentence that is true in $\mathfrak{N}$, the standard structure, but not provable from the axioms of N. Our sentence, θ, will "say" that θ is not provable from N, and in order to "say" that, we will need a formula that will identify the (Gödel numbers of the) formulas that *are* provable from N. To do that we will need to be able to code up deductions from N, which makes it necessary to code up sequences of formulas. Thus, our next goal will be to settle on a coding scheme for sequences of $\mathcal{L}_{NT}$-formulas.

We have been pretty careful with our coding up to this point. If you check, every Gödel number that we have used has been even, with the exception of 3, which is the garbage case in Definition 5.7.1. We will now use numbers with smallest prime factor 5 to code sequences of formulas.

Suppose that we have the sequence of formulas

$$D = (\phi_1, \phi_2, \ldots, \phi_k).$$

We will define the **sequence code of** D to be the number

$$\ulcorner D \urcorner = 5^{\ulcorner \phi_1 \urcorner} 7^{\ulcorner \phi_2 \urcorner} \cdots p_{k+2}^{\ulcorner \phi_k \urcorner}.$$

So the exponent on the $(i+2)^{\text{nd}}$ prime is the Gödel number of the i^{th} element of the sequence. You are asked in the Exercises to produce several useful $\mathcal{L}_{NT}$-formulas relating to sequence codes.

We will be interested in using sequence codes to code up deductions from N. If you look back at the definition of a deduction (Definition 2.2.1), you will see that to check if a sequence is a deduction, we need only check that each entry is either an axiom or follows from previous lines of the deduction via a rule of inference. So to say that c codes up a deduction from N, we want to be able to say, for each entry e at position i of the deduction coded by c,

$$\mathit{AxiomOfN}(e) \vee \mathit{LogicalAxiom}(e) \vee \mathit{RuleOfInference}(c, e, i).$$

The first two of these we have already developed. The last major goal of this section (and this chapter) is to flesh out the details of a Δ-definition of a formula that recognizes when an entry in a deduction is justified by one of our rules of inference.

As you recall from Section 2.4, there are two types of rules of inference: propositional consequence and quantifier rules. The latter of these is easiest to Δ-define, so we deal with it first.

The rule of inference (QR) is Definition 2.4.6, which says that if x is not free in ψ, then the following are rules of inference:

$$\big(\{\psi \to \phi\}, \psi \to (\forall x \phi)\big)$$
$$\big(\{\phi \to \psi\}, (\exists x \phi) \to \psi\big).$$

If we look at the first of these, we see that if e is an entry in a code for a deduction, and $e = \ulcorner \psi \to (\forall x \phi) \urcorner$, then e is justified as long as there is an earlier entry in the deduction that codes up the formula $\psi \to \phi$, assuming that x is not free in ψ. So all we have to do is figure out a way to say this.

We will write the formula $\mathit{QRRule1}(c, e, i)$, where c is the code of the deduction, and e is the entry at position i that is being justified by the first quantifier rule. In the following, f is playing the role of $\ulcorner \psi \urcorner$, while g is supposed to be $\ulcorner \phi \urcorner$:

$QRRule1(c, e, i)$ is:

$$SequenceCode(c) \wedge IthSequenceElement(e, i, c) \wedge$$
$$(\exists x, f, g < c)\Big[Formula(f) \wedge Formula(g) \wedge Variable(x) \wedge$$
$$\neg Free(x, f) \wedge$$
$$e = \overline{2}^{\overline{4}}\overline{3}^{\overline{2}^{\overline{2}}\overline{3}^{Sf}+1}\overline{5}^{\overline{2}^{\overline{6}}\overline{3}^{Sx}\overline{5}^{Sg}+1} \wedge$$
$$(\exists j < i)(\exists e_j < c)\big(IthSequenceElement(e_j, j, c) \wedge$$
$$e_j = \overline{2}^{\overline{4}}\overline{3}^{\overline{2}^{\overline{2}}\overline{3}^{Sf}+1}\overline{5}^{Sg}\big)\Big].$$

After you write out *QRRule2* in the Exercises, it is obvious to define

$QRRule(c, e, i)$ is:

$$QRRule1(c, e, i) \vee QRRule2(c, e, i).$$

Now we have to address propositional consequence. This will involve some rather tricky coding, so hold on tight as we review propositional logic.

Assume that D is our deduction, and D is the sequence of formulas

$$(\alpha_1, \alpha_2, \alpha_3, \ldots, \alpha_k).$$

Notice that entry α_i of a deduction is justified as a propositional consequence if and only if the formula

$$\beta = (\alpha_1 \wedge \alpha_2 \wedge \cdots \wedge \alpha_{i-1}) \rightarrow \alpha_i$$

is a tautology.

Now, as we discussed in Section 2.4, in order to decide if a first-order formula β is a tautology, we must take β and find the propositional formula β_P. Then if β_P is

$$\Big[(\alpha_1)_P \wedge (\alpha_2)_P \wedge \cdots \wedge (\alpha_{i-1})_P\Big] \rightarrow (\alpha_i)_P,$$

we must show any truth assignment that makes $(\alpha_1)_P$ through $(\alpha_{i-1})_P$ true must also make $(\alpha_i)_P$ true.

As outlined on page 51, to create a propositional version of a formula, we first must find the prime components of that formula, where a prime component is a subformula that is either universal and not contained in any other universal subformula, or atomic and not contained in any universal formula. Rather than explicitly writing out a Δ-formula that identifies the pairs (u, v) such that u is the Gödel number of a prime component of the formula with Gödel number v, let us write down a recursive definition of

this set of formulas, and we will leave it to the reader to find the minor modification of Theorem 5.10.2, which will guarantee that this set of Gödel numbers is representable.

Definition 5.12.1. If β and γ are $\mathcal{L}_{NT}$-formulas, γ is said to be a **prime component** of β if:

1. β is atomic and $\gamma = \beta$, or

2. β is universal and $\gamma = \beta$, or

3. β is $\neg\alpha$ and γ is a prime component of α, or

4. β is $\alpha_1 \vee \alpha_2$ and γ is a prime component of either α_1 or α_2.

Proposition 5.12.2. *The set*

$$\begin{aligned}&\textsc{PrimeComponent} = \\ &\quad \{(u, f) \mid u = \ulcorner\gamma\urcorner \textit{ and } f = \ulcorner\beta\urcorner \textit{ and } \gamma \textit{ is a prime component of } \beta, \\ &\qquad\qquad \textit{for some } \mathcal{L}_{NT}\textit{-formulas } \gamma \textit{ and } \beta\}\end{aligned}$$

is representable and has Δ*-definition* $\boxed{\mathit{PrimeComponent}(u, f)}$.

Proof. Theorem 5.10.2. □

Now we will code up a canonical sequence of all of the prime components of α_1 through α_i. We will say r codes the PrimeList for the first i entries of the deduction coded by c if each element coded by r is a prime component of one of the first i entries of the deduction coded by c, r's elements are distinct, each prime component of each of the first i entries of the deduction coded by c is among the entries in r, and if s is a smaller code number, s is missing one of these prime components:

$$
\begin{aligned}
&\mathit{PrimeList}(c,i,r) \text{ is:}\\
&\quad \mathit{SequenceCode}(c) \wedge \mathit{CodeNumber}(r) \wedge\\
&\quad (\exists l < r)\Big[\mathit{Length}(r,l) \wedge\\
&\quad (\forall m,n \le l)(\forall e_m, e_n < r)\\
&\quad (\mathit{IthElement}(e_m,m,r) \wedge \mathit{IthElement}(e_n,n,r)) \rightarrow\\
&\quad \Big((\exists k \le i)(\exists f_k \le c)\mathit{IthSequenceElement}(f_k,k,c) \wedge\\
&\quad \mathit{PrimeComponent}(e_m,f_k) \wedge\\
&\quad [(m \ne n) \rightarrow (e_m \ne e_n)]\Big) \wedge\\
&\quad \big[(\forall k \le i)(\forall f_k \le c)(\forall u \le f_k)(\mathit{IthSequenceElement}(f_k,k,c) \wedge\\
&\quad \mathit{PrimeComponent}(u,f_k) \rightarrow\\
&\quad (\exists m < l)\mathit{IthElement}(u,m,r))\big]\Big] \wedge\\
&\quad (\forall s < r)\Big(\mathit{CodeNumber}(s) \rightarrow\\
&\quad (\exists k \le i)(\exists f_k \le c)(\exists u \le f_k)\big[\mathit{IthSequenceElement}(f_k,k,c) \wedge\\
&\quad \mathit{PrimeComponent}(u,f_k) \wedge\\
&\quad (\forall m < s)(\neg \mathit{IthElement}(u,m,s))\big]\Big)
\end{aligned}
$$

To decide if β is a tautology, we need to assign all possible truth values to all of the prime components in our list, and then evaluate the truth of each α_i under a given truth assignment. So the next thing we need to do is find a way to code up an assignment of truth values to all of the prime components of all the α_i's. We will say that v codes up a truth assignment if v is the code number of a sequence of the right length and all of the elements coded by v are either 0 (for false) or 1 (for true).

$$
\begin{aligned}
&\mathit{TruthAssignment}(c,i,r,v) \text{ is:}\\
&\quad \mathit{PrimeList}(c,i,r) \wedge \mathit{CodeNumber}(v) \wedge\\
&\quad (\exists l < r)\big(\mathit{Length}(r,l) \wedge \mathit{Length}(v,l) \wedge\\
&\quad (\forall i \le l)(\forall e < v)(\mathit{IthElement}(e,i,v) \rightarrow\\
&\quad [e = \overline{0} \vee e = \overline{1}])\big).
\end{aligned}
$$

Now, given a truth assignment for the prime components of α_1 through α_i, coded up in v, we need to be able to evaluate the truth of each formula

α_n under that assignment. To do this we will need to be able to evaluate the truth value of a single formula.

Suppose that we first look at an example. Here is a formula construction sequence that ends with some formula α:

$$\big(0 < x, x < y, (0 < x \vee x < y), \neg(0 < x), \\ (\forall x)(0 < x \vee x < y), \underbrace{(\forall x)(0 < x \vee x < y) \vee (\neg 0 < x)}_{\alpha}\big).$$

Let us assume that the PrimeList we are working with is

$$((\forall x)(0 < x \vee x < y), 0 < x)$$

and the chosen truth assignment for our PrimeList is

$$(0.1).$$

To assign the truth value to α, we follow along the construction sequence. When we see an entry that is in the PrimeList, we assign that entry the corresponding truth value from our truth assignment. If the entry in the construction sequence is not a prime component, one of three things might be true:

1. The entry might be universal, in which case we assign it truth value 2 (for *undefined*).
2. The entry might be an atomic formula that ends up inside the scope of a quantifier in α. Again, we use truth value 2.
3. The entry might be the denial of or disjunction of earlier entries in the construction sequence. In this case we can figure out its truth value, always using 2 if any of the parts have truth value 2.

So to continue our example from above, the sequence of truth values would be

$$(1, 2, 2, 0, 0, 0).$$

Exercise 7 asks you to write a Δ-formula $\boxed{\mathit{Evaluate}(e, r, v, y)}$, where you should assume that e is the Gödel number for a formula α, r is a code for a list including all of the prime components of α, v is a TruthAssignment for r, and y is the truth value for α, given the truth assignment v. Exercise 8 also concerns this formula.

Now, knowing how to evaluate the truth of a single formula α, we will be able to decide if α_i, the i^{th} element of the alleged deduction that is coded by c, can be justified by the propositional consequence rule. Recall that we need only check whether

$$(\alpha_1 \wedge \alpha_2 \wedge \cdots \wedge \alpha_{i-1}) \rightarrow \alpha_i$$

is a tautology. To do this, we need only see whether any truth assignment that makes α_1 through α_{i-1} true also makes α_i true. Here is the Δ-formula that says this, where c codes the alleged deduction, and e is supposed to be the code for the i^{th} entry in the deduction, the entry that is to be justified by an appeal to the rule PC:

$$
\begin{array}{l}
PCRule(c,e,i) \text{ is:} \\
\qquad IthSequenceElement(e,i,c) \wedge \\
\qquad (\exists r < \left[\overline{2}^{\left(c^{\overline{2}}\right)^{\left(c^{\overline{2}}\right)}}\right]^{c^{\overline{3}}}) \\
\qquad (\forall v < \left(\left[\overline{2}^{\left(c^{\overline{2}}\right)^{\left(c^{\overline{2}}\right)}}\right]^{\overline{\ulcorner\overline{1}\urcorner}}\right)^{c^{\overline{2}}}) \\
\qquad \Big(\big[PrimeList(c,i,r) \wedge TruthAssignment(c,i,r,v)\big] \rightarrow \\
\qquad\qquad \Big[\big((\forall j < i)(\exists e_j < c) \\
\qquad\qquad (IthSequenceElement(e_j,j,c) \wedge Evaluate(e_j,r,v,\overline{1}))\big) \\
\qquad\qquad\qquad \rightarrow \\
\qquad\qquad\qquad\qquad Evaluate(e,r,v,\overline{1})\Big]\Big).
\end{array}
$$

Now, to know that this works, we must justify the bounds that we have given for r and v. The number r is supposed to code the list of prime components among the first i elements of the deduction c. So r is of the form $5^{\ulcorner\gamma_1\urcorner}7^{\ulcorner\gamma_2\urcorner}\cdots p_{k+2}^{\ulcorner\gamma_k\urcorner}$, where the prime components are γ_1 through γ_k. First, we need to get a handle on the number of prime components there are. Since c is the code for the deduction, there are fewer than c formulas in the deduction, and each of those formulas has a Gödel number that is less than or equal to c. So each formula in the deduction has fewer than c symbols in it, and thus fewer than c prime components. So we have no more than c formulas, each with no more than c prime components, so there are no more than c^2 prime components total in the deduction coded by c. If we look at the number r, we see that

$$
\begin{aligned}
r &= 5^{\ulcorner\gamma_1\urcorner}7^{\ulcorner\gamma_2\urcorner}\cdots p_{k+2}^{\ulcorner\gamma_k\urcorner} \\
&\leq 5^c 7^c \cdots p_{k+2}^c \\
&\leq 5^c 7^c \cdots p_{c^2}^c \\
&\leq \left[2^{(c^2)^{(c^2)}}\right]^{c^3},
\end{aligned}
$$

where the last line depends on our usual bound for the size of the (c^2)th prime, as we saw on page 146.

As for v, v is a code number that gives us the truth values of the various prime components coded in r. Thus v is of the form $2^{i_1}3^{i_2}\cdots p_k^{i_k}$, where there are k prime components, and each i_j is either $\ulcorner\overline{0}\urcorner$ or $\ulcorner\overline{1}\urcorner$. By the argument above, we know there are no more than c^2 prime components, so

$$\begin{aligned} v &= 2^{i_1}3^{i_2}\cdots p_k^{i_k} \\ &\leq 2^{\ulcorner\overline{1}\urcorner}3^{\ulcorner\overline{1}\urcorner}\cdots p_{c^2}^{\ulcorner\overline{1}\urcorner} \\ &\leq \left(\left[2^{(c^2)^{(c^2)}}\right]^{\ulcorner\overline{1}\urcorner}\right)^{c^2}. \end{aligned}$$

Thus we have justified the bounds given for r and v in the definition of $PCRule(c,e,i)$. Thus we have a Δ-definition, and the set PCRule is representable.

Now we have to remember where we were on page 160. We are thinking of c as coding an alleged deduction from N, and we need to check all of the entries of c to see if they are legal. We have already written Δ-formulas *LogicalAxiom* and *AxiomOfN*, and we are working on the rules of inference. The quantifier rules were relatively easy, so we then wrote a Δ-formula $PCRule(c,e,i)$ that is true if and only if e is the i^{th} entry of the deduction c and can be justified by the rule PC.

At last, we are able to decide if c is the code for a deduction from N of a formula with Gödel number f:

$$\begin{aligned} &Deduction(c,f) \text{ is:} \\ &\qquad SequenceCode(c) \wedge Formula(f) \wedge \\ &(\exists l < c)\Big(SequenceLength(c,l) \wedge IthSequenceElement(f,l,c) \wedge \\ &\qquad (\forall i \leq l)(\forall e < c)\Big[IthSequenceElement(e,i,c) \rightarrow \\ &\qquad\qquad \Big(LogicalAxiom(e) \vee AxiomOfN(e) \vee \\ &\qquad\qquad\qquad QRRule(c,e,i) \vee PCRule(c,e,i) \Big)\Big]\Big). \end{aligned}$$

This formula represents the set Deduction $\subseteq \mathbb{N}^2$ and shows that Deduction is a representable set. Given Church's Thesis, this makes formal the ideas of Chapter 2, where it was suggested that we ought to be able to program a computer to decide if an alleged deduction is, in fact, a deduction. We said earlier that if a computer could decide whether an

alleged axiom was an axiom, in other words, if the collection of axioms was representable, and whether a use of a rule of inference was legitimate, then it could check whether an alleged deduction was ok. This brings us to a slightly sticky point.

You will notice that we have carefully been using bounded quantification and that the formula $Deduction(c, f)$ above is a Δ-formula. But that depends on the fact that the collection of axioms N has a Δ-definition, which is more restrictive than just saying that the collection of axioms is a representable set. So perhaps, if A is a representable set of axioms that does not have a Δ-definition, we have a problem. Fortunately, this is not the case.

Suppose that A is a representable set of axioms. We know, via Exercises 13 and 14 of Section 5.3, that A has a numeralwise determined definition $AxiomOfA(e)$. Furthermore, if we define the formula $Deduction_A$ as above, replacing $AxiomOfN$ with $AxiomOfA$, then $Deduction_A$ is a numeralwise determined formula that defines the set DEDUCTION_A, and so the collection of codes of deductions from our representable set of axioms A is itself representable and our computer can, in fact, check whether an alleged deduction from A is, in fact, a deduction from A.

If you keep the equivalence between "computer-decidable" and representable in your head, we will say more about this in Chapters 6 and 7.

5.12.1 Exercises

1. Write a Δ-formula $\boxed{SequenceCode(c)}$ that is true in $\mathfrak{N}$ if and only if c is the code of a sequence of $\mathcal{L}_{NT}$-formulas.

2. Write a Δ-formula $\boxed{SequenceLength(c, l)}$ that is true in $\mathfrak{N}$ if and only if c is a sequence code of a sequence of length l.

3. Write out a Δ-formula $\boxed{IthSequenceElement(e, i, c)}$ that is true in $\mathfrak{N}$ if and only if c is a sequence code and the i^{th} element of the sequence coded by c has Gödel number e.

4. Write out a Δ-formula $QRRule2(c, e, i)$ that will be true in $\mathfrak{N}$ if e is the i^{th} entry in the sequence coded by c and is justified by the second quantifier rule.

5. Here is your average, ordinary tautology:

$$\phi(x, y) \text{ is } \big[[\forall x P(x)] \rightarrow (Q(x, y) \rightarrow [\forall x P(x)])\big].$$

Find a construction sequence for ϕ. Make a list of the prime components of ϕ. Pretending that your list of prime components is *the* prime list for ϕ, find all possible truth assignments for ϕ and use the truth assignments to evaluate the truth of ϕ under your assignments. If all goes well, every time you evaluate the truth of ϕ, you will get: True.

6. Repeat Exercise 5 with the following formulas, which are not (necessarily) guaranteed to be tautologies:

   (a) $(\forall x)(x < y \rightarrow x < y)$

   (b) $(\forall x)(x < y) \rightarrow (\forall x)(x < y)$

   (c) $(A(x) \vee B(y)) \vee \neg B(x)$

   (d) $(A(x) \vee B(y)) \rightarrow \big[(\forall x)(\forall y)(A(x) \vee B(y))\big]$

   (e) $\big[(\forall x)(\forall y)(A(x) \vee B(y))\big] \rightarrow (A(x) \vee B(y))$

7. Write out the Δ-formula $\mathit{Evaluate}(e, r, v, y)$, as outlined in the text. You will need to think about formula construction sequences, as you did in Exercise 6 in Section 5.8.1.

8. Show by induction on the longest of the two construction sequences that if d_1 and d_2 are codes for two construction sequences of α, and if $\mathit{Evaluate}(d_1, r, v, y_1)$ and $\mathit{Evaluate}(d_2, r, v, y_2)$ are both true, then y_1 and y_2 are equal. Thus, the truth assigned to α does not depend upon the construction sequence chosen to evaluate that truth.

5.13 Summing Up, Looking Ahead

Well, in all likelihood you are exhausted at this point. This chapter has been full of dense, technical arguments with imposing definition piled upon imposing definition. We have established our axioms, discussed representable sets, and talked about Δ-definitions. You have just finished wading through an unending stream of Δ-definitions that culminated with the formula $\mathit{Deduction}(c, f)$ which holds if and only if c is a code for a deduction of the formula with Gödel number f. We have succeeded in coding up our deductive theory inside of number theory.

Let us reiterate this. If you look at that formula *Deduction*, what it looks like is a disjunction of a lot of equations and inequalities. Everything is written in the language $\mathcal{L}_{NT}$, so *everything* in that formula is of the form $SSS0 < SS0 + x$ (with, it must be admitted, rather more S's than shown here). Although we have given these formulas names which suggest that they are about formulas and terms and tautologies and deductions, the formulas are formulas of elementary number theory, so the formulas don't know that they are about anything beyond whether this number is bigger than that number, no matter how much you want to anthropomorphize them. The interpretation of the numbers as standing for formulas via the scheme of Gödel numbering is imposed on those numbers by us.

The next chapter brings us to the statement and the proof of Gödel's Incompleteness Theorem. To give you a taste of things to come, notice that if we define the statement

$Thm_N(f)$ is:

$$(\exists c)(Deduction(c, f)),$$

then $Thm_N(f)$ should hold if and only if f is the Gödel number of a formula that is a theorem of N. We are sure that you noticed that Thm_N is not a Δ-formula, and there is no way to fix that—we cannot bound the length of a deduction of a formula. But Thm_N *is* a Σ-formula, and Proposition 5.3.13 tells us that true Σ-sentences are provable. That will be one of the keys to Gödel's proof.

Well, if 90% of the iceberg is under water, we've covered that. Now it is time to examine that glorious 10% that is left.

Chapter 6

The Incompleteness Theorems

6.1 Introduction

Suppose that A is a collection of axioms in the language of number theory such that A is consistent and is simple enough so that we can decide whether or not a given formula is an element of A. The First Incompleteness Theorem will produce a sentence, θ, such that $\mathfrak{N} \models \theta$ and $A \nvdash \theta$, thus showing our collection of axioms A is incomplete.

The idea behind the construction of θ is really neat: We get θ to say that θ is not provable from the axioms of A. In some sense, θ is no more than a fancy version of the Liar's Paradox, in which the speaker asserts that the speaker is lying, inviting the listener to decide whether that utterance is a truth or a falsehood. The challenge for us is to figure out how to get an $\mathcal{L}_{NT}$-sentence to do the asserting!

You will notice that there are two parts to θ. The first is that θ will have to talk about the collection of Gödel numbers of theorems of A. That is no problem, as we will have a Σ-formula $Thm_A(f)$ that is true (and thus provable from N) if and only if f is the Gödel number of a theorem of A. The thing that makes θ tricky is that we want θ to be $Thm_A(\overline{a})$, where $a = \ulcorner\theta\urcorner$. In this sense, we need θ to refer to itself. Showing that we can do that will be the content of the Self-Reference Lemma that we address in the next section.

After proving the First Incompleteness Theorem, we will discuss some corollaries and improvements to the theorem before moving on to discuss the Second Incompleteness Theorem, which states that the set of axioms of Peano Arithmetic cannot prove that Peano Arithmetic is consistent, unless (of course) Peano Arithmetic is inconsistent, in which case it can prove anything. So our goal of proving that we have a complete, consistent set

of axioms for $\mathfrak{N}$ is a goal that cannot be reached within the confines of first-order logic.

6.2 The Self-Reference Lemma

Our goal in this section is to show that, given any formula with only one free variable, we can construct a sentence that asserts that the given formula applies to itself. Before we do that, however, we need a lemma.

You will certainly(!) recall from Section 5.9 that we have the representable function Num : $\mathbb{N} \to \mathbb{N}$ such that $\text{Num}(n) = \ulcorner \overline{n} \urcorner$. We have a Δ-formula, $Num(x, y)$ that represents the representable relation Num. We would love to know that N is strong enough to prove, for example, the $\mathcal{L}_{NT}$-formula

$$Num(\overline{3}, y) \leftrightarrow y = \overline{\text{Num}(3)}.$$

> *Chaff:* You weren't confused by the use of Num(3), were you? It might be confusing since Num is a set, and Num $\subseteq \mathbb{N}^2$. But we know that Num is a function and so the thing that is named Num(3) is the unique element y of the the codomain $\mathbb{N}$ such that the ordered pair $(3, y) \in$ Num. That was obvious, right?

Unfortunately, although wanting the displayed equivalence to be provable in N is eminently reasonable, we cannot quite do it. The problem is that our formula *Num* is not quite tricky enough. The lemma that we will state will give this result for any representable *function,* so we will state it in that generality. For our purposes, however, it will be enough to know that it applies to the representable functions Num and Sub of Section 5.9.

Lemma 6.2.1. *Suppose that* $\text{R} \subseteq \mathbb{N}^{n+1}$ *is a representable set represented (in N) by the* $\mathcal{L}_{NT}$*-formula R. If* R *is a function with domain* $\mathbb{N}^n$ *and codomain* $\mathbb{N}$*, then there is a formula Rf such that*

1. *Rf represents* R*, and*
2. *for any* $a_1, \ldots, a_n \in \mathbb{N}$*,*

$$N \vdash \left(Rf(\overline{a}_1, \ldots, \overline{a}_n, y) \leftrightarrow y = \overline{\text{R}(a_1, \ldots, a_n)} \right).$$

Proof. To improve the readability, we assume that $n = 1$. Let the relation R be a function with domain $\mathbb{N}$, and let the formula Rf be defined by

$$Rf(x, y) :\equiv R(x, y) \wedge (\forall i < y)[\neg R(x, i)].$$

We first prove (1), that Rf represents the set R. As we already know that R represents R, it suffices to prove that $N \vdash R(\overline{a}, \overline{b})$ iff $N \vdash Rf(\overline{a}, \overline{b})$.

Assume that $N \vdash Rf(\overline{a}, \overline{b})$. Then, we have $N \vdash R(\overline{a}, \overline{b})$ by our rule of inference (PC).

For the converse, assume that $N \vdash R(\overline{a}, \overline{b})$. Since R is a function, we have $(a, i) \notin \mathrm{R}$ for all $i < b$. Thus, since R represents R, we have

$$N \vdash \neg R(\overline{a}, \overline{0}) \ \wedge \ \neg R(\overline{a}, \overline{1}) \ \wedge \ \ldots \ \wedge \ \neg R(\overline{a}, \overline{b-1}) \ .$$

By Corollary 5.3.12, this means that $N \vdash (\forall i < \overline{b})[\neg R(x, i)]$. Therefore, $N \vdash Rf(\overline{a}, \overline{b})$, establishing (1).

We now turn to the proof of (2). First we prove that $N \vdash Rf(\overline{a}, y) \rightarrow y = \overline{\mathrm{R}(a)}$. Since R is a function and R represents R, we have

$$N \vdash \neg R(\overline{a}, \overline{0}) \ \wedge \ \ldots \ \wedge \ \neg R(\overline{a}, \overline{\mathrm{R}(a) - 1}) \ .$$

(Be careful with the typeface there—remember the difference between the formula R and the function R.)

Again by Corollary 5.3.12, we have $N \vdash (\forall y < \overline{\mathrm{R}(a)})[\neg R(\overline{a}, y)]$. Thus, we also have

$$N \vdash \Big(y < \overline{\mathrm{R}(a)} \rightarrow \neg R(\overline{a}, y) \Big) . \tag{i}$$

By (i) and (PC), we have

$$N \vdash \Big(R(\overline{a}, y) \rightarrow \neg y < \overline{\mathrm{R}(a)} \Big) . \tag{ii}$$

By the logical axioms, we know that

$$\vdash \Big[(\forall i < y)[\neg R(\overline{a}, i)] \ \rightarrow \ \big(\overline{\mathrm{R}(a)} < y \ \rightarrow \ \neg R(\overline{a}, \overline{\mathrm{R}(a)}) \big) \Big] . \tag{iii}$$

In addition, since the formula R represents the function R, we have

$$N \vdash R(\overline{a}, \overline{\mathrm{R}(a)}). \tag{iv}$$

By (iii), (iv) and (PC), we have

$$N \vdash \Big((\forall i < y)[\neg R(\overline{a}, i)] \ \rightarrow \ \neg \overline{\mathrm{R}(a)} < y \Big) . \tag{v}$$

By (ii), (v) and (PC), we have

$$N \vdash \big(\ R(\overline{a}, y) \ \wedge \ (\forall i < y)[\neg R(\overline{a}, i)] \ \big) \ \rightarrow \ \big(\ \neg y < \overline{\mathrm{R}(a)} \wedge \ \neg \overline{\mathrm{R}(a)} < y \ \big). \tag{vi}$$

By (vi) and the axiom N11, we have

$$N \vdash \big(\ R(\overline{a}, y) \ \wedge \ (\forall i < y)[\neg R(\overline{a}, i)] \ \big) \ \rightarrow \ y = \overline{\mathrm{R}(a)}),$$

that is, $N \vdash \Big(Rf(\overline{a}, y) \rightarrow y = \overline{\mathrm{R}(a)} \Big)$, which establishes the forward direction of our biconditional.

To complete the proof of (2), we also need to prove that

$$N \vdash \Big(y = \overline{\mathrm{R}(a)} \rightarrow Rf(\overline{a}, y) \Big) .$$

This is left for the reader as Exercise 1. □

Now, to the Self-Reference Lemma. This is just a lovely result, insightful in its concept and far reaching in its consequences. We'd love to say that the proof was also lovely and enlightening, but to be honest, we don't have an enlightening sort of proof to show you. Sometimes the best way to describe a proof is that the argument sort of picks you up and shakes you until you agree that it does, in fact, establish what it is supposed to establish. That's what you get here. So, get ready for a technical argument and some intricate calculations, but keep in mind that the result, key to establishing the Incompleteness Theorem, really is quite pretty.

Lemma 6.2.2 (Gödel's Self-Reference Lemma). *Let $\psi(v_1)$ be an $\mathcal{L}_{NT}$-formula with only v_1 free. Then there is a sentence ϕ such that*

$$N \vdash \left(\phi \leftrightarrow \psi(\overline{\ulcorner \phi \urcorner})\right).$$

Chaff: Look at how neat this is! Do you see how ϕ "says" *ψ is true of me*? And we can do this for any formula ψ! What a cool idea!

Proof. We will explicitly construct the needed ϕ. Recall that in Section 5.9 we defined representable functions $\text{Num} : \mathbb{N} \to \mathbb{N}$ and $\text{Sub} : \mathbb{N}^3 \to \mathbb{N}$ such that $\text{Num}(n) = \ulcorner \overline{n} \urcorner$ and $\text{Sub}(\ulcorner \alpha \urcorner, \ulcorner x \urcorner, \ulcorner t \urcorner) = \ulcorner \alpha^x_t \urcorner$. By Lemma 6.2.1 we know that there are formulas *Numf* and *Subf* such that

$$N \vdash \Big[\mathit{Numf}(\overline{a}, y) \leftrightarrow y = \overline{\text{Num}(a)}\Big], \text{ and that}$$
$$N \vdash \Big[\mathit{Subf}(\overline{a}, \overline{b}, \overline{c}, z) \leftrightarrow z = \overline{\text{Sub}(a, b, c)}\Big].$$

Chaff: Remember, Num is the function and *Numf* is an $\mathcal{L}_{NT}$-formula that represents the function! Oh, and just because we're going to need it, $\ulcorner v_1 \urcorner = 8$.

Now suppose that $\psi(v_1)$ is given as in the statement of the lemma. Let $\gamma(v_1)$ be

$$\forall y \forall z \Big[\big[\mathit{Numf}(v_1, y) \wedge \mathit{Subf}(v_1, \overline{8}, y, z)\big] \to \psi(z)\Big].$$

Let us look at $\gamma(n)$ a little more closely, supposing that $n = \ulcorner \alpha \urcorner$. If the antecedent of $\gamma(n)$ holds, then the first part of the antecedent tells us that

$$y = \text{Num}(n) = \ulcorner \overline{n} \urcorner$$

and the second part of the antecedent asserts that

$$\begin{aligned} z &= \text{Sub}(n, 8, \ulcorner \overline{n} \urcorner) \\ &= \text{Sub}(\ulcorner \alpha \urcorner, \ulcorner v_1 \urcorner, \ulcorner \overline{n} \urcorner) \\ &= \ulcorner \alpha^{v_1}_{\overline{n}} \urcorner \\ &= \ulcorner \alpha^{v_1}_{\overline{\ulcorner \alpha \urcorner}} \urcorner. \end{aligned}$$

So z is the Gödel number of {α with the Gödel number of α substituted in for v_1}.

One more tricky choice will get us to where we want to go. Let $m = \ulcorner\gamma(v_1)\urcorner$, and let ϕ be $\gamma(\overline{m})$. Certainly, ϕ is a sentence, so we will be finished if we can show that $N \vdash \phi \leftrightarrow \psi(\overline{\ulcorner\phi\urcorner})$.

Let us work through a small calculation first. Notice that

$$\begin{aligned}\mathrm{Sub}(m, 8, \ulcorner\overline{m}\urcorner) &= \mathrm{Sub}(\ulcorner\gamma(v_1)\urcorner, \ulcorner v_1\urcorner, \ulcorner\overline{m}\urcorner)\\ &= \ulcorner\gamma(v_1)^{v_1}_{\overline{m}}\urcorner\\ &= \ulcorner\gamma(\overline{m})\urcorner\\ &= \ulcorner\phi\urcorner. \qquad (6.1)\end{aligned}$$

With this in hand, the following are provably equivalent in N:

ϕ

$\forall y\forall z\Big[Numf(\overline{m}, y) \to \big(Subf(\overline{m}, \overline{8}, y, z) \to \psi(z)\big)\Big]$ logic

$\forall y\forall z\Big[y = \overline{\mathrm{Num}(m)} \to \big(Subf(\overline{m}, \overline{8}, y, z) \to \psi(z)\big)\Big]$ Lemma 6.2.1

$\forall y\forall z\Big[y = \overline{\ulcorner\overline{m}\urcorner} \to \big(Subf(\overline{m}, \overline{8}, y, z) \to \psi(z)\big)\Big]$ calculation

$\forall z\Big(Subf(\overline{m}, \overline{8}, \overline{\ulcorner\overline{m}\urcorner}, z) \to \psi(z)\Big)$ quantifier rules

$\forall z\Big(z = \overline{\mathrm{Sub}(m, 8, \ulcorner\overline{m}\urcorner)} \to \psi(z)\Big)$ Lemma 6.2.1

$\forall z\Big(z = \overline{\ulcorner\phi\urcorner} \to \psi(z)\Big)$ calculation (6.1) above

$\psi(\overline{\ulcorner\phi\urcorner})$ quantifier rules

So $N \vdash \phi \leftrightarrow \psi\Big(\overline{\ulcorner\phi\urcorner}\Big)$, as needed. □

Notice in this proof that if ψ is a Π-formula, then ϕ is logically equivalent to a Π-sentence. By altering γ slightly we can also arrange, if ψ is a Σ-formula, to have ϕ logically equivalent to a Σ-sentence.

6.2.1 Exercises

1. Complete the proof of Claim (2) of Lemma 6.2.1 by showing that

$$N \vdash \Big(y = \overline{\mathrm{R}(a)} \to Rf(\overline{a}, y)\Big).$$

2. The proof of Lemma 6.2.1 depended on the use of the corollary to Rosser's Lemma, Corollary 5.3.12. To make the reading easier, we assumed in the proof that $n = 1$, which made the use of the corollary much easier. Work through the proof of Lemma 6.2.1 assuming that $n = 2$, being careful about the details.

3. Suppose that $\psi(v_1)$ is $Formula(v_1)$. By the Self-Reference Lemma, there is a sentence ϕ such that $N \vdash \big(\phi \leftrightarrow Formula(\overline{\ulcorner \phi \urcorner})\big)$. Does $N \vdash \phi$? Does $N \vdash \neg\phi$? Justify your answer. What happens if we use $\psi(v_1) = \neg Formula(v_1)$ instead?

4. Let $\psi(v_1)$ be $Even(v_1)$, and let ϕ be the sentence generated when the Self-Reference Lemma is applied to $\psi(v_1)$. Does $N \vdash \phi$? Does $N \vdash \neg\phi$? How can you tell?

5. Show that the proof of the Self-Reference Lemma still works if we use

$$\gamma(v_1) = \exists y \exists z \Big[Numf(v_1, y) \wedge Subf(v_1, \overline{8}, y, z) \wedge \psi(z) \Big].$$

Conclude that if ψ is a Σ-formula, then the ϕ of the Self-Reference Lemma can be taken to be equivalent to a Σ-sentence.

6.3 The First Incompleteness Theorem

We are ready to state and prove the First Incompleteness Theorem, which tells us that if we are given any reasonable axiom system A, there is a sentence that is true in $\mathfrak{N}$ but not provable from A.

You may have been complaining all along about my choice for an axiom system. Perhaps you have been convinced from the beginning that N is clearly too weak to prove every truth about the natural numbers. You are right. We know, for example, that N can neither prove nor refute the commutative law of addition. Since you are a diligent person, we imagine that you have come up with an axiom system of your own, let's call it A. You might be convinced that A is the "right" choice of axioms, a set of axioms that is strong enough to prove every truth about $\mathfrak{N}$. Although this shows admirable independence on your part, we will, unfortunately, be able to prove that your set of axioms A also fails to be complete, as long as A satisfies certain reasonable conditions.

Definition 6.3.1. A **theory** is a collection of formulas T that is closed under deduction: For every formula ϕ, if $T \vdash \phi$, then $\phi \in T$. If A is a set of formulas, then the **theory of** A, written $Th(A)$, is the smallest theory that includes A: $Th(A) = \{\sigma \mid A \vdash \sigma\}$. If $\mathfrak{A}$ is an $\mathcal{L}$-structure, the **theory of** $\mathfrak{A}$, written $Th(\mathfrak{A})$ is the set of formulas that are true in the structure: $Th(\mathfrak{A}) = \{\sigma \mid \mathfrak{A} \models \sigma\}$.

Definition 6.3.2. A theory T in the language $\mathcal{L}_{NT}$ is said to be **axiomatized** by a set of formulas A if $T = Th(A)$.

If, in addition, the set $\textsc{AxiomOfA} \underset{\text{def}}{=} \{\ulcorner \alpha \urcorner \mid \alpha \in A\}$ is a representable set, we say that T is **recursively axiomatized**.

> *Chaff:* A bit of notation. We will say sets *of* $\mathcal{L}_{NT}$*-formulas* are recursive or not recursive. Sets of *numbers* or *functions* mapping natural numbers to natural numbers will be representable or not representable. In Chapter 7 we will introduce recursive functions, but that's later.

Roughly, a theory is recursively axiomatized if the theory has an axiom set that is simple enough so that we can recognize axioms and nonaxioms. One of the conditions that we will require of your set of axioms A is that it be recursive.

> *Chaff:* In general, if we say that some set of formulas F has a property, like being representable, or definable, or cute, or whatever, that should only apply to natural numbers, we are really saying that the set $\{\ulcorner\alpha\urcorner \mid \alpha \in F\}$ is representable/definable/cute. We trust that you will be able to keep confusion at bay.

We will prove just a couple of more lemmas before stating and proving the Incompleteness Theorem. First, we will show that any representable set is Σ-definable. We mentioned this back in Section 5.3 when we first mentioned that a set with a Δ-definition had to be representable.

Lemma 6.3.3. *Suppose that $A \subseteq \mathbb{N}$ is representable. Then A is Σ-definable. In other words, there is a Σ-formula $\phi(v_1)$ such that*

$$a \in A \text{ if and only if } \mathfrak{N} \models \phi(\overline{a}).$$

Proof. Let A be given and assume that A is representable. Then we know that there is a formula with one free variable, $\psi(v_1)$, such that ψ represents A:

$$a \in A \text{ if and only if } N \vdash \psi(\overline{a}).$$

Let k be the Gödel number of this formula, so $k = \ulcorner\psi(v_1)\urcorner$ and consider

$$\phi(v_1) \text{ is } \exists z \exists y \exists c \left(Num(v_1, z) \wedge Sub(\overline{k}, \overline{8}, z, y) \wedge Deduction(c, y) \right).$$

Since *Num*, *Sub*, and *Deduction* are all Δ-formulas (we talked about them in the last section, and they were defined back in Sections 5.9 and 5.12), certainly ϕ is a Σ-formula, so we need only to argue that $a \in A$ if and only if $\phi(\overline{a})$ is true in the standard model.

Let's first assume that $a \in A$. Then, we know that if $z = \ulcorner\overline{a}\urcorner$, then $\mathfrak{N} \models Num(a, z)$. (There should be bars over the a and the z, but let's leave them off so that we can read things more naturally.) If, in addition $y = \ulcorner\psi(\overline{a})\urcorner$, then since $8 = \ulcorner v_1 \urcorner$ we know that $\mathfrak{N} \models Sub(k, 8, z, y)$. So all we need to do is show that there is some code for a deduction (in N) of the formula with Gödel number y, which means that we must show that there

is a deduction-in-N of $\psi(\overline{a})$. But we have assumed that ψ is a formula that represents A, and since $a \in A$, we know that $N \vdash \psi(\overline{a})$. If we just let c be any code of any deduction of $\psi(\overline{a})$, we see that $\mathfrak{N} \models \mathit{Deduction}(c, y)$, and thus that $\mathfrak{N} \models \phi(\overline{a})$.

For the converse, assume that $\mathfrak{N} \models \phi(\overline{a})$. We must prove that $a \in A$. As $\phi(a)$ is true in the standard model, there is some natural number c that codes a deduction for the formula with Gödel number y, and by the definitions of *Num* and *Sub*, we know that $y = \ulcorner \psi(\overline{a}) \urcorner$. But this means that c is a code for a deduction-in-N of $\psi(\overline{a})$, and therefore $N \vdash \psi(\overline{a})$. But this means, as $\psi(v_1)$ represents A, that $a \in A$, as needed. □

Recall that if A is a set of formulas, then $Th(A)$, the theory of A, is the collection of formulas that A can prove, the theorems of A. We will be interested in the set of Gödel numbers of the formulas that are elements of $Th(A)$.

Definition 6.3.4. If A is a set of formulas, then

$$\textsc{Thm}_A = \{\ulcorner \phi \urcorner \mid A \vdash \phi\}.$$

Now, let's assume for a minute that you have a recursive set of axioms A. It turns out that not only is A Σ-definable, but $\textsc{Thm}_A$ is Σ-definable, as well.

Lemma 6.3.5. *If A is a recursive set of formulas, then $\textsc{Thm}_A$ is Σ-definable.*

Proof. Suppose that A is recursive, in other words that AxiomOfA is representable. We must find a Σ-formula that defines $\textsc{Thm}_A$.

As AxiomOfA is representable, by Lemma 6.3.3, we know that there is a Σ-formula $\mathit{AxiomOfA}(x)$ that defines AxiomOfA.

We define the formula $\mathit{Deduction}_A(c, v_1)$ by copying the definition of $\mathit{Deduction}(c, f)$ from page 165, replacing the f's by v_1's and replacing the $\mathit{AxiomOfN}(e)$ with $\mathit{AxiomOfA}(e)$. Notice that, unlike $\mathit{Deduction}(c, f)$, $\mathit{Deduction}_A(c, v_1)$ is not a Δ-formula, but rather a Σ-formula.

Now let the formula $\mathit{Thm}_A(v_1)$ be $\exists c \mathit{Deduction}_A(c, v_1)$. Then $\mathit{Thm}_A(v_1)$ is a Σ-formula that defines (notice, not represents—defines) the set $\textsc{Thm}_A$, as you can readily check. □

Enough with the lemmas! Gödel proved, in his First Incompleteness Theorem, that any collection of $\mathcal{L}_{NT}$-axioms A that is reasonably simple (recursive) and consistent must be incomplete. We will show that there is a sentence that is true in the standard model $\mathfrak{N}$ that is not proven by the axioms of A.

Theorem 6.3.6 (Gödel's First Incompleteness Theorem). *Suppose that A is a consistent and recursive set of axioms in the language $\mathcal{L}_{NT}$. Then there is a sentence θ such that $\mathfrak{N} \models \theta$ but $A \nvdash \theta$.*

Proof. We assume that A is strong enough to prove all of the axioms of N. If not, any unprovable-from-A axiom of N will be a sentence that is true in $\mathfrak{N}$ and unprovable from A.

With this assumption, use the Self-Reference Lemma 6.2.2 and the Σ-formula $Thm_A(v_1)$ of the proof of Lemma 6.3.5 to produce a sentence θ such that

$$N \vdash \left[\theta \leftrightarrow \neg Thm_A\left(\overline{\ulcorner \theta \urcorner}\right)\right].$$

Chaff: Do you see how θ "says" *I am not a theorem*?

Now $\mathfrak{N} \models \left[\theta \leftrightarrow \neg Thm_A\left(\overline{\ulcorner \theta \urcorner}\right)\right]$, so we know that

$$\begin{aligned} \mathfrak{N} \models \theta \text{ iff } \mathfrak{N} &\not\models Thm_A\left(\overline{\ulcorner \theta \urcorner}\right) && \text{Definition of satisfaction} \\ \text{iff } \ulcorner \theta \urcorner &\notin \textsc{Thm}_A && Thm_A \text{ defines } \textsc{Thm}_A \\ \text{iff } A &\nvdash \theta, && \text{Definition of } \textsc{Thm}_A \end{aligned}$$

so θ is either true in $\mathfrak{N}$ and not provable from A, or false in $\mathfrak{N}$ and provable from A.

Assume, for the moment, that θ is false and $A \vdash \theta$. Then $\ulcorner \theta \urcorner \in \textsc{Thm}_A$, so $\mathfrak{N} \models Thm_A(\overline{\ulcorner \theta \urcorner})$ as Thm_A defines $\textsc{Thm}_A$. But then $Thm_A(\overline{\ulcorner \theta \urcorner})$ is a true Σ-sentence, and so by Proposition 5.3.13, this means that $N \vdash Thm_A\left(\overline{\ulcorner \theta \urcorner}\right)$. This means, by our choice of θ, $N \vdash \neg\theta$. Since A proves all of the axioms of N, this implies that $A \vdash \neg\theta$. But we already have assumed that $A \vdash \theta$, which means that A is inconsistent, contrary to our assumption on A.

Thus, $A \nvdash \theta$ and $\mathfrak{N} \models \theta$, and θ is true and unprovable, as needed.

□

> *Chaff:* We've pulled together a lot of material here into a pretty compact argument. Look it over again, and make sure that you see how we have used the recursiveness of our set of axioms A and then the Σ-definability of $\textsc{Thm}_A$, to bring the proof to a close.

Much of the rest of this chapter will focus on taking Gödel's Incompleteness Theorem and refining it. We'll start by doing a little more work on the front end and getting an estimate of how complicated the Gödel sentence θ has to be.

Theorem 6.3.7. *Suppose that A is a consistent and recursive set of axioms in the language $\mathcal{L}_{NT}$. Then there is a Π-sentence θ such that $\mathfrak{N} \models \theta$ but $A \nvdash \theta$.*

Proof. We just have to keep track of what we did in the last proof. As before, if A cannot prove all of the axioms of N, then whichever N-axiom A does not prove is a Π-sentence that is true in $\mathfrak{N}$ and not provable by A.

So we can assume that A is strong enough to prove the axioms of N. Using the sentence θ that we constructed in the proof of Theorem 6.3.6 and looking at $\neg Thm_A\left(\overline{\ulcorner\theta\urcorner}\right)$, we see that it is logically equivalent (via DeMorgan's Laws) to a Π-sentence θ_Π. So, via the Completeness Theorem

$$\vdash \left[\theta_\Pi \leftrightarrow \neg Thm_A\left(\overline{\ulcorner\theta\urcorner}\right)\right].$$

By our choice of θ, and by the fact that A is strong enough to prove the axioms of N, we know that

$$A \vdash \left[\theta \leftrightarrow \neg Thm_A\left(\overline{\ulcorner\theta\urcorner}\right)\right].$$

So $A \vdash \theta$ if and only if $A \vdash \theta_\Pi$. Since we know that $A \nvdash \theta$, we know that $A \nvdash \theta_\Pi$.

On the semantic side of things, since θ_Π and $\neg Thm_A\left(\overline{\ulcorner\theta\urcorner}\right)$ are logically equivalent, $\mathfrak{N} \models \theta_\Pi$ if and only if $\mathfrak{N} \models \neg Thm_A\left(\overline{\ulcorner\theta\urcorner}\right)$. Since we just finished proving in Theorem 6.3.6 that θ, and hence $\neg Thm_A\left(\overline{\ulcorner\theta\urcorner}\right)$ are true in $\mathfrak{N}$, we can conclude that $\mathfrak{N} \models \theta_\Pi$.

Therefore θ_Π is a Π-sentence that is true in the standard model but not provable from our axioms A.

□

Gödel's Theorem gives us an example of a set that is not representable:

Corollary 6.3.8. *If A is a consistent, recursive set of axioms in the language $\mathcal{L}_{NT}$ that proves the axioms of N, then* THM_A *is not representable.*

Proof. Suppose that THM_A is representable. Then some formula $\gamma(v_1)$ represents the set in N, which means that

$$\text{If } f \in \text{THM}_A, \text{ then } N \vdash \gamma(\overline{f}).$$
$$\text{If } f \notin \text{THM}_A, \text{ then } N \vdash \neg\gamma(\overline{f}).$$

Now, using the Self-Reference Lemma, construct a sentence θ such that

$$N \vdash \left[\theta \leftrightarrow \neg\gamma\left(\overline{\ulcorner\theta\urcorner}\right)\right].$$

Assume for the moment that $A \vdash \theta$. As THM_A is representable, this would mean that $N \vdash \gamma\left(\overline{\ulcorner\theta\urcorner}\right)$, which implies that $N \vdash \neg\theta$. So $\mathfrak{N} \models \neg\theta$, and θ is false in $\mathfrak{N}$.

On the other hand, if $A \nvdash \theta$, then $N \vdash \neg\gamma\left(\overline{\ulcorner\theta\urcorner}\right)$, and so $N \vdash \theta$, implying that θ is true in $\mathfrak{N}$.

These two comments lead us to conclude that θ is either true in $\mathfrak{N}$ and not provable from A, or false in $\mathfrak{N}$ and provable from A.

Now our argument closely follows the proof of the First Incompleteness Theorem. If θ were false and provable, then $\mathfrak{N} \models Thm_A(\ulcorner\theta\urcorner)$, and so

$\ulcorner\theta\urcorner \in \text{THM}_A$ and by the definition of γ, we know $N \vdash \gamma\left(\overline{\ulcorner\theta\urcorner}\right)$. But now our construction of θ leads us to conclude that $N \vdash \neg\theta$. As A is sufficiently strong to prove the axioms of N, this tells us that $A \vdash \neg\theta$, which contradicts the assumption that A is consistent.

But, if we assume that θ is true and unprovable, then as $\gamma(v_1)$ represents THM_A and $\ulcorner\theta\urcorner \notin \text{THM}_A$, $N \vdash \neg\gamma\left(\overline{\ulcorner\theta\urcorner}\right)$. By our choice of θ, this means that N, and hence A, proves θ, contradicting the assumption that θ is not provable from A.

So our assumption that THM_A is representable leads to a contradiction, and we are led to conclude that THM_A is not representable. □

> *Chaff:* This corollary is the "computers will never put mathematicians out of a job" corollary: If you accept the identification between representable sets and sets for which a computer can decide membership, Corollary 6.3.8 says that we will never be able to write a computer program which will accept as input an $\mathcal{L}_{NT}$-formula ϕ and will produce as output "ϕ is a theorem" if $A \vdash \phi$ and "ϕ is not a theorem" if $A \nvdash \phi$.
>
> If you think of the computer as taking ϕ and systematically listing all deductions and checking to see if it has listed a deduction-of-ϕ, it is easy to see that if ϕ is, in fact, a theorem-of-A, the computer will eventually verify that fact. If, however, ϕ is not a theorem, the computer will never know. All the computer will know is that it has not succeeded, as of this moment, of finding a deduction of ϕ. But it will not be able to say that it will never come across a deduction of ϕ.

We can, in fact, dispense with the requirement that A be a recursive set of axioms:

Theorem 6.3.9. *Suppose that A is a consistent set of axioms extending N and in the language $\mathcal{L}_{NT}$. Then the set THM_A is not representable in A.*

Proof. Suppose, to the contrary, that $\gamma(v_1)$ represents THM_A. As usual, let θ be such that

$$N \vdash \left[\theta \leftrightarrow \neg\gamma\left(\overline{\ulcorner\theta\urcorner}\right)\right].$$

As A extends N, certainly

$$A \vdash \left[\theta \leftrightarrow \neg\gamma\left(\overline{\ulcorner\theta\urcorner}\right)\right].$$

Now

$$\begin{aligned}
A \vdash \theta &\Rightarrow \ulcorner\theta\urcorner \in \textsc{Thm}_A \\
&\Rightarrow A \vdash \gamma\left(\overline{\ulcorner\theta\urcorner}\right) && \text{since } \gamma \text{ represents } \textsc{Thm}_A \\
&\Rightarrow A \vdash \neg\theta && \text{choice of } \theta \\
&\Rightarrow A \nvdash \theta && A \text{ is consistent} \\
&\Rightarrow \ulcorner\theta\urcorner \notin \textsc{Thm}_A \\
&\Rightarrow A \vdash \neg\gamma\left(\overline{\ulcorner\theta\urcorner}\right) && \text{since } \gamma \text{ represents } \textsc{Thm}_A \\
&\Rightarrow A \vdash \theta && \text{choice of } \theta.
\end{aligned}$$

This contradiction completes the proof. □

The short version of the Theorem 6.3.9 is: Any consistent theory extending N is undecidable, where "undecidable" means not recursive.

Let us apply Theorem 6.3.9 to a particular theory, the theory of the natural numbers, $Th(\mathfrak{N})$.

Theorem 6.3.10 (Tarski's Theorem). *The set of Gödel numbers of formulas true in $\mathfrak{N}$ is not definable in $\mathfrak{N}$.*

Proof. Recall that for any set $A \subseteq \mathbb{N}$, to say ϕ defines A in $\mathfrak{N}$ means that

$$\begin{aligned}
&\text{If } a \in A, \text{ then } \mathfrak{N} \models \phi(\overline{a}). \\
&\text{If } a \notin A, \text{ then } \mathfrak{N} \models \neg\phi(\overline{a}).
\end{aligned}$$

Since $Th(\mathfrak{N}) \vdash \alpha$ if and only if $\mathfrak{N} \models \alpha$, we can rewrite this statement as

$$\begin{aligned}
&\text{If } a \in A, \text{ then } Th(\mathfrak{N}) \vdash \phi(\overline{a}). \\
&\text{If } a \notin A, \text{ then } Th(\mathfrak{N}) \vdash \neg\phi(\overline{a}).
\end{aligned}$$

So ϕ defines a set in $\mathfrak{N}$ if and only if ϕ represents the set in $Th(\mathfrak{N})$. The set in question is

$$\begin{aligned}
&\textsc{TrueIn}\mathfrak{N} = \\
&\qquad \{a \mid a \text{ is the Gödel number of a formula that is true in } \mathfrak{N}\}.
\end{aligned}$$

Notice that this is precisely the set

$$\begin{aligned}
&\textsc{Thm}_{Th(\mathfrak{N})} = \\
&\qquad \{a \mid a \text{ is the Gödel number of a formula provable from } Th(\mathfrak{N})\},
\end{aligned}$$

so $\textsc{TrueIn}\mathfrak{N}$ is definable if and only if $\textsc{Thm}_{Th(\mathfrak{N})}$ is representable in $Th(\mathfrak{N})$. But $Th(\mathfrak{N})$ is a consistent set of axioms extending N, so by Theorem 6.3.9, $\textsc{Thm}_{Th(\mathfrak{N})}$ is not representable in $Th(\mathfrak{N})$. So $\textsc{TrueIn}\mathfrak{N}$ is not definable. □

Chaff: Tarski's Theorem has an easy shorthand version:

Truth is undefinable.

Throw that one around when you are having a deep philosophical discussion with your friends or when you are trying to explain to your relatives what you've been learning in your advanced logic studies.

As you look back over this section, you will notice that in our statement of the Incompleteness Theorem, we demanded that the set of axioms A be consistent and recursive. If we examine those assumptions a little more carefully, we might learn something. To ask that A be consistent seems necessary, for if A is not consistent, then A is strong enough to prove every formula, and so A is (trivially) complete.

On the other hand, you might want to see if it would be possible to relax the requirement that A be recursive. For example, maybe we could define a set of axioms A by some formula and then prove that A is complete. Just to show that this isn't going to be trivial, we have the following.

Proposition 6.3.11. *Suppose that A is a consistent and Σ-definable set of axioms in the language $\mathcal{L}_{NT}$. Then there is a sentence θ such that $\mathfrak{N} \models \theta$ but $A \nvdash \theta$.*

Proof. If you read through the proof of Theorem 6.3.6 you will notice that the only way we used the recursiveness of A was in noting that Thm_A was Σ-definable, which we knew from Lemma 6.3.3. But in the proof of Lemma 6.3.3 we only needed to know that there was a Σ-formula $AxiomOfA(x)$ that defined the set AxiomOfA. The existence of that formula does not require A to be recursive, only Σ-definable, so the assumptions of this proposition are strong enough to carry through the proof of Gödel's Incompleteness Theorem, even if A is only known to be Σ-definable. □

6.3.1 Exercises

1. Prove the theorem that is implicit in Definition 6.3.1: If A is a set of formulas, then $\{\sigma \mid A \vdash \sigma\}$ is a theory.

2. Suppose that $\mathfrak{A}$ is an $\mathcal{L}$-structure, and consider $Th(\mathfrak{A})$, as defined in Definition 3.3.4. Prove that $Th(\mathfrak{A})$ is a theory in the sense of Definition 6.3.1.

3. Assume that $A \vdash N$. The First Incompleteness Theorem in the version of Theorem 6.3.7 gives us a Π-sentence θ such that $\mathfrak{N} \models \theta$ and $A \nvdash \theta$. Can we find a Σ-sentence with the same characteristics? Please justify your answer.

6.4 Extensions and Refinements of Incompleteness

If you look carefully at the First Incompleteness Theorem, it does not quite say that the collection of axioms A is incomplete. All that is claimed is that there is a sentence θ such that θ is true-in-$\mathfrak{N}$ and θ is not provable from A. But, perhaps, $\neg\theta$ *is* provable from A. Our first result in this section brings the focus onto incompleteness.

Proposition 6.4.1. *Suppose that A is a consistent, recursive set of axioms that proves all of the axioms of N. If all of the axioms of A are true in $\mathfrak{N}$, then there is a sentence θ such that $A \nvdash \theta$ and $A \nvdash \neg\theta$.*

Proof. As in the proof of the First Incompleteness Theorem, let θ be such that

$$N \vdash \left[\theta \leftrightarrow \neg Thm_A\left(\overline{\ulcorner\theta\urcorner}\right)\right].$$

We know that $\mathfrak{N} \models \theta$ and $A \nvdash \theta$. Suppose that A proves $\neg\theta$. Then, as all of the axioms of A are true in the structure $\mathfrak{N}$, we know that $\mathfrak{N} \models \neg\theta$, which contradicts the fact that $\mathfrak{N} \models \theta$. Thus $A \nvdash \neg\theta$, and A is incomplete. □

We can also eliminate the hypothesis that the axioms of A be true in $\mathfrak{N}$. To prove that A is incomplete, all that will be required of our axioms will be that they form a consistent extension of N. We will prove this result in two steps, starting by strengthening the hypothesis of consistency to ω-consistency. Then, in Rosser's Theorem, we will show how a slightly trickier use of the Self-Reference Lemma can show that any consistent, recursive extension of N must be incomplete.

Definition 6.4.2. A theory T in $\mathcal{L}_{NT}$ is said to be **ω-inconsistent** if there is a formula $\phi(x)$ such that $T \vdash \exists x\phi(x)$, but for each natural number n, $T \vdash \neg\phi(\overline{n})$. Otherwise, T is called **ω-consistent**.

Proposition 6.4.3. *If T is ω-consistent, then T is consistent.*

Proof. Exercise 2. □

The converse of this proposition is false, as you are asked to show in Exercise 3. Also notice that if T is a theory such that $\mathfrak{N} \models T$, then T is necessarily ω-consistent. So the chain of implications looks like this:

$$\text{true in } \mathfrak{N} \Rightarrow \omega\text{-consistent} \Rightarrow \text{consistent}.$$

We already know that if A is a recursive, true-in-$\mathfrak{N}$ extension of N, then A is incomplete. In Proposition 6.4.4 we will use the same sentence θ as in the First Incompleteness Theorem to show that an ω-consistent recursive extension of N is incomplete, then in Theorem 6.4.5 we will use a slightly trickier sentence ρ to show that mere consistency suffices: Any consistent, recursive extension of N must be incomplete.

Proposition 6.4.4. *If A is an ω-consistent and recursive set of axioms extending N, then A is incomplete.*

Proof. As usual, let θ be such that $N \vdash \left[\theta \leftrightarrow \neg \mathit{Thm}_A\left(\overline{\ulcorner \theta \urcorner}\right)\right]$. We already know that $\mathfrak{N} \models \theta$ and $A \nvdash \theta$. We will show that $A \nvdash \neg\theta$, and thus A is incomplete.

Assume that $A \vdash \neg\theta$; then we know by our choice of θ that $A \vdash \mathit{Thm}_A(\overline{\ulcorner \theta \urcorner})$. In other words,

$$A \vdash (\exists x)\mathit{Deduction}_A(x, \overline{\ulcorner \theta \urcorner}). \tag{6.2}$$

Since we also know that $A \nvdash \theta$, we know, for each natural number n, that n is not the code for a deduction of θ. So

$$\text{For each } n \in \mathbb{N},\ (n, \ulcorner \theta \urcorner) \notin \textsc{Deduction}_A,$$

and thus, as the formula $\mathit{Deduction}_A$ represents the set $\textsc{Deduction}_A$,

$$\text{For each } n \in \mathbb{N},\ N \vdash \neg \mathit{Deduction}_A(\overline{n}, \overline{\ulcorner \theta \urcorner}).$$

Since A is an extension of N, we have

$$\text{For each } n \in \mathbb{N},\ A \vdash \neg \mathit{Deduction}_A(\overline{n}, \overline{\ulcorner \theta \urcorner}),$$

which, when combined with (6.2), implies that A is ω-inconsistent, contrary to hypothesis. So our assumption must be wrong, and $A \nvdash \neg\theta$, as needed. □

We have gotten a lot of mileage out of our sentence θ, but J. Barkley Rosser used a different sentence to get a stronger result.

Theorem 6.4.5 (Rosser's Theorem). *If A is a set of $\mathcal{L}_{NT}$-axioms that is recursive, consistent, and extends N, then A is incomplete.*

Proof. We use the Self-Reference Lemma to construct a sentence ρ such that

$$N \vdash \Big[\rho \leftrightarrow (\forall x)\big[\mathit{Deduction}_A(x, \overline{\ulcorner \rho \urcorner}) \rightarrow (\exists y < x)(\mathit{Deduction}_A(y, \overline{2}^{\overline{2}}\overline{3}^{\overline{\ulcorner \rho \urcorner + 1}}))\big]\Big].$$

So ρ says, "If there is a proof of ρ, then there is a proof of $\neg\rho$ with a smaller code."

First, we claim that $A \nvdash \rho$: Assume, on the contrary, that $A \vdash \rho$. Let a be a number that codes up a deduction of ρ. Since the formula $\mathit{Deduction}_A$ represents the set $\textsc{Deduction}_A$, we know that

$$A \vdash \mathit{Deduction}_A(\overline{a}, \overline{\ulcorner \rho \urcorner}).$$

Also, by choice of ρ, we know that

$$A \vdash (\forall x)\big[\mathit{Deduction}_A(x, \overline{\ulcorner \rho \urcorner}) \to (\exists y < x)(\mathit{Deduction}_A(y, \overline{2}^{\overline{2}}\overline{3}^{\overline{\ulcorner \rho \urcorner + 1}}))\big],$$

which means that

$$A \vdash \big[\mathit{Deduction}_A(\overline{a}, \overline{\ulcorner \rho \urcorner}) \to (\exists y < \overline{a})(\mathit{Deduction}_A(y, \overline{2}^{\overline{2}}\overline{3}^{\overline{\ulcorner \rho \urcorner + 1}}))\big].$$

But we know that $A \vdash \mathit{Deduction}_A(\overline{a}, \overline{\ulcorner \rho \urcorner})$, so we are led to conclude that

$$A \vdash (\exists y < \overline{a})(\mathit{Deduction}_A(y, \overline{2}^{\overline{2}}\overline{3}^{\overline{\ulcorner \rho \urcorner + 1}})). \tag{6.3}$$

On the other hand, we have assumed that $A \vdash \rho$ and A is consistent. Therefore, we know, for each $n \in \mathbb{N}$, that

$$A \vdash \neg \mathit{Deduction}_A(\overline{n}, \overline{2}^{\overline{2}}\overline{3}^{\overline{\ulcorner \rho \urcorner + 1}}).$$

But then, by Rosser's Lemma (Lemma 5.3.11), we know that

$$A \vdash \neg(\exists y < \overline{a})(\mathit{Deduction}_A(y, \overline{2}^{\overline{2}}\overline{3}^{\overline{\ulcorner \rho \urcorner + 1}})),$$

which, when combined with (6.3), shows that A is inconsistent, a contradiction. So we conclude that $A \nvdash \rho$, as claimed.

We also claim that $A \nvdash \neg\rho$. For assume that b is a code for a deduction of $\neg\rho$. Then $A \vdash \mathit{Deduction}_A(\overline{b}, \overline{\ulcorner \neg \rho \urcorner})$. In other words, $A \vdash \mathit{Deduction}_A(\overline{b}, \overline{2}^{\overline{2}}\overline{3}^{\overline{\ulcorner \rho \urcorner + 1}})$.

Now, since $A \vdash \neg\rho$, from the choice of ρ we know that

$$A \vdash (\exists x)\left[\mathit{Deduction}_A(x, \overline{\ulcorner \rho \urcorner}) \wedge \neg(\exists y)[y < x \wedge \mathit{Deduction}_A(y, \overline{2}^{\overline{2}}\overline{3}^{\overline{\ulcorner \rho \urcorner + 1}})]\right].$$

So if we substitute $\overline{b}$ for y, we see that

$$A \vdash (\exists x)\left[\mathit{Deduction}_A(x, \overline{\ulcorner \rho \urcorner}) \wedge \neg[\overline{b} < x \wedge \underbrace{\mathit{Deduction}_A(\overline{b}, \overline{2}^{\overline{2}}\overline{3}^{\overline{\ulcorner \rho \urcorner + 1}})}_{(*)}]\right].$$

But as the formula (*) is provable from A, this means that

$$A \vdash (\exists x)\big[\mathit{Deduction}_A(x, \overline{\ulcorner \rho \urcorner}) \wedge \neg(\overline{b} < x)\big],$$

which is equivalent to

$$A \vdash (\exists x)\big[x \leq \overline{b} \wedge \mathit{Deduction}_A(x, \overline{\ulcorner \rho \urcorner})\big].$$

Now, since we have assumed that $A \vdash \neg\rho$ and A is consistent, we know that for each $n \in \mathbb{N}$, $A \vdash \neg \mathit{Deduction}_A(\overline{n}, \overline{\ulcorner \rho \urcorner})$, and so by Rosser's Lemma again,

$$A \vdash \neg(\exists x)\big[x \leq \overline{b} \wedge \mathit{Deduction}_A(x, \overline{\ulcorner \rho \urcorner})\big],$$

which shows that A is inconsistent, contrary to assumption. Therefore, $A \nvdash \neg\rho$.

Since $A \nvdash \rho$ and $A \nvdash \neg\rho$, we know that A is incomplete, as needed. $\square$

6.4.1 Exercises

1. Suppose that A and B are theories in some language and $A \subseteq B$. Suppose that B is consistent. Show that A is consistent. What happens if B is ω-consistent?

2. Prove Proposition 6.4.3. [*Suggestion:* Try the contrapositive.]

3. Find an example of a theory that is consistent but not ω-consistent. [*Suggestion:* If you can construct the correct sort of model, $\mathfrak{A}$, then $Th(\mathfrak{A})$ will be consistent and ω-inconsistent.]

4. Suppose θ is such that

$$N \vdash \left[\theta \leftrightarrow Thm_N\left(\overline{\ulcorner \neg \theta \urcorner}\right)\right].$$

So θ asserts its own refutability. Is θ true? Provable? Refutable?

6.5 Another Proof of Incompleteness

As we mentioned in the introduction to this chapter, the sentence θ of the First Incompleteness Theorem can be seen as a formalization of the liar paradox, where a speaker asserts that what the speaker says is false. In this section we will outline a proof, due to George Boolos [Boolos 94] of the First Incompleteness Theorem that is based upon Berry's paradox.

G. G. Berry, a librarian at Oxford University at the beginning of the twentieth century, is credited by Bertrand Russell with the observation that *the least integer not nameable in fewer than nineteen syllables* is nameable in eighteen syllables. We will formalize a version of Berry's phrase to come up with another sentence that is true in $\mathfrak{N}$ but not provable.

For our argument to work we need to make a minor change in our language. It will be important that our language have only finitely many symbols, and to make $\mathcal{L}_{NT}$ finite, we have to rework the way that we denote variables. So, for this section, rather than having *Vars* be the infinite set of variables $v_1, v_2, \ldots, v_n, \ldots$, and thinking of each v_i as its own symbol, we will think of them as a sequence of symbols. So the string v_{17} is no longer a single symbol but is, rather, three symbols. The set *Vars* then is defined to be the collection of finite strings of symbols that are of the form v_s, where s is a string of digits. Thus the symbols of $\mathcal{L}_{NT}$ are

$$\{(,), \vee, \neg, \forall, =, v, 0, 1, 2, \ldots, 9, 0, S, +, \cdot, E, <\},$$

giving us precisely 23 symbols in the language, and $\mathcal{L}_{NT}$ is finite.

We restate the First Incompleteness Theorem:

Theorem 6.5.1. *Suppose that Q is a consistent and recursive set of $\mathcal{L}_{NT}$-formulas. Then there is a sentence β such that $\mathfrak{N} \models \beta$ and $Q \nvdash \beta$.*

Outline of Proof. We can assume that Q proves all of the axioms of N, since if not, one of the axioms of N would do for β.

Suppose that $\phi(x)$ is a formula of $\mathcal{L}_{NT}$. We say that $\phi(x)$ **names the natural number** n **with respect to the axioms** Q if and only if

$$Q \vdash \big((\forall x)(\phi(x) \leftrightarrow x = \overline{n})\big).$$

Notice that no formula can name more than one number.

Here is where we use the fact that our language is finite. Fix a number i. Since there are only 23 symbols in our language, there are no more that 23^i formulas of length i, where the length of a formula is the number of symbols that it contains. So no more than 23^i numbers can be named by formulas of length i.

So for each number m, there are only finitely many numbers that can be named by formulas of length less than or equal to m. Thus, for each m, there are some numbers that cannot be named by formulas of length less than or equal to m, so there is a *least* number not named by any formula containing no more than m symbols.

Now there is a formula $\eta(v_1, l)$ with two free variables that says that v_1 is a number named by a formula of length l. You are asked in Exercise 5 to find η. Then we can let $\delta(v_1, v_2)$ be $(\exists l < v_2)\eta(v_1, l)$. Thus $\delta(v_1, v_2)$ says that the number v_1 is nameable in fewer than v_2 symbols.

Two more formulas get us home. Define $\gamma(v_1, v_2)$ by

$$\gamma(v_1, v_2) \text{ is } \big[\neg\delta(v_1, v_2)\big] \wedge \big[(\forall x < v_1)\delta(x, v_2)\big].$$

So $\gamma(v_1, v_2)$ says that v_1 is the least number not nameable in fewer than v_2 symbols. Let k be the number of symbols in $\gamma(v_1, v_2)$. Notice that $k > 4$. (How's that for a bit of an understatement?)

We can now define

$$\alpha(v_1) \text{ is } (\exists v_2)(v_2 = \overline{10} \cdot \overline{k} \wedge \gamma(v_1, v_2)).$$

So $\alpha(v_1)$ claims that v_1 is the least number not nameable in fewer than $10k$ symbols, where k is the number of symbols in $\gamma(v_1, v_2)$. If we write out $\alpha(v_1)$ formally, we see that

$$\alpha(v_1) \text{ is } (\neg(\forall v_2)(\neg(= v_2 \cdot \overline{10}\,\overline{k} \wedge \gamma(v_1, v_2)))).$$

Let us count the symbols in $\alpha(v_1)$. There are 11 symbols in $\overline{10}$, $k+1$ in $\overline{k}$, and k in $\gamma(v_1, v_2)$. By using both my fingers and my toes, I get a total of $11 + (k+1) + k + 18 = 2k + 30$ symbols in $\alpha(v_1)$. A bit of algebra tells us that since $k > 4$, $10k > 2k + 30$, so $\alpha(v_1)$ contains fewer than $10k$ symbols.

Suppose that b is the smallest number not named by any formula with fewer than $10k$ symbols. Since $\alpha(v_1)$ has fewer than $10k$ symbols, certainly b is not named by the formula $\alpha(v_1)$. By looking back at our definition of what it means for a formula to name a number, we see that

$$Q \nvdash \big((\forall v_1)(\alpha(v_1) \leftrightarrow v_1 = \overline{b})\big).$$

But this is where we want to be. Let β be the sentence

$$\big((\forall v_1)(\alpha(v_1) \leftrightarrow v_1 = \overline{b})\big).$$

We just saw that Q does not prove the sentence β. But β is a true statement about the natural numbers, for the number b *is* the least number that is not nameable in fewer than $10k$ symbols, and that is precisely the meaning of the sentence β. So $\mathfrak{N} \models \beta$ and $Q \nvdash \beta$, as needed. □

One big difference between this proof and our first proof of the First Incompleteness Theorem is that this proof does not use the Self-Reference Lemma, as we don't have to substitute the Gödel number of a formula into itself, but rather, we substitute the numeral of a number that makes the formula true. But both proofs do rely on Gödel numbering and coding deductions, so the mechanism of Chapter 5 comes into play for both.

6.5.1 Exercises

1. Give an argument, perhaps based on Church's Thesis or perhaps using a variant of Theorem 5.10.2, to show that there is a Δ-formula $\mathit{LengthOfFormula}(f, l)$ that is true if and only if f is the Gödel number of a formula consisting of exactly l symbols. [*Suggestion:* You may need to start by showing the existence of a formula *LengthOfTerm* with two free variables.]

2. Prove that no formula can name two different numbers. [*Suggestion:* Think about what it would mean if $\phi(x)$ named both 17 and 42.]

3. Find an upper bound for the number of numbers that can be named by formulas $\phi(x)$ that contain no more than m symbols.

4. Suppose that $\phi(x)$ names n with respect to the set of axioms N. Show that $\phi(x)$ represents $\{n\}$.

5. Find a formula $\eta(n, l)$ that says that n is named by a formula of length l. Is your formula equivalent to a Σ-formula?

6. The statement of Theorem 6.5.1 makes two assumptions about the collection of axioms Q. Where are they used in the proof?

6.6 Peano Arithmetic and the Second Incompleteness Theorem

It is our goal in this section to show that a set of axioms cannot prove its own consistency. Now this statement needs to be sharpened, for of course *some* sets of axioms can prove their own consistency. For example, the

axiom set A might contain the statement "A is consistent." But that will lead to problems, as we will show.

The first order of business will be to introduce a new collection of axioms, called PA, or the axioms of Peano Arithmetic. This extension of N will be recursive and will be true in $\mathfrak{N}$, so all of the results of this chapter will apply to PA. We will then state, without proof, three properties that are true of PA, properties that are needed for the proof of the Second Incompleteness Theorem.

The Second Incompleteness Theorem is, in some sense, nothing more than finding another true and unprovable statement, but the statement that we will find is much more natural and has a longer history than the sentence θ of Gödel I. As we mentioned on page 1, at the beginning of the twentieth century, the German mathematician David Hilbert proposed that the mathematical community set itself the goal of proving that mathematics is consistent. In the Second Incompleteness Theorem, we will see that no extension of Peano Arithmetic can prove itself to be consistent, and thus certainly any plan for a self-contained proof of consistency must be doomed to failure. This was the blow that Gödel delivered to Hilbert's consistency program. Understanding the ideas behind this second proof is our current goal. We will not fill in all of the details of the construction. The interested reader is directed to Craig Smoryński's article in [Barwise 77], on which our presentation is based.

We begin by establishing our new set of nonlogical axioms, the axioms of Peano Arithmetic.

Definition 6.6.1. The axioms of **Peano Arithmetic**, or PA, are the eleven axioms of N together with the axiom schema

$$\Big[\phi(0) \wedge (\forall x)\big[\phi(x) \to \phi(Sx)\big]\Big] \to (\forall x)\phi(x)$$

for each $\mathcal{L}_{NT}$-formula ϕ with one free variable.

So Peano Arithmetic is nothing more than the familiar set of axioms N, together with an induction schema for $\mathcal{L}_{NT}$-definable sets. Although we will not go through the details, it is not difficult to see that the set AxiomOfPA is representable, so PA is a recursively axiomatized extension of N.

What makes PA useful to us is that PA is strong enough to prove certain facts about derivations in PA. In particular, PA is strong enough so that the following derivability conditions hold for all formulas ϕ:

$$\text{If } PA \vdash \phi, \text{ then } PA \vdash \mathit{Thm}_{PA}(\overline{\ulcorner \phi \urcorner}). \tag{D1}$$

$$PA \vdash \Big[\mathit{Thm}_{PA}(\overline{\ulcorner \phi \urcorner}) \to \mathit{Thm}_{PA}\big(\overline{\ulcorner \mathit{Thm}_{PA}(\overline{\ulcorner \phi \urcorner})\urcorner}\big)\Big]. \tag{D2}$$

$$PA \vdash \Big[\big[\mathit{Thm}_{PA}(\overline{\ulcorner \phi \urcorner}) \wedge \mathit{Thm}_{PA}(\overline{\ulcorner \phi \to \psi \urcorner})\big] \to \mathit{Thm}_{PA}(\overline{\ulcorner \psi \urcorner})\Big]. \tag{D3}$$

Granting these conditions, we can move on to prove the Second Incompleteness Theorem. Recall that we agreed to use the symbol $\perp$ for the contradictory sentence $\big[(\forall x)x = x\big] \wedge \neg\big[(\forall x)x = x\big]$.

Definition 6.6.2. The sentence Con_{PA} is the sentence $\neg Thm_{PA}(\overline{\ulcorner \perp \urcorner})$.

Notice that $\mathfrak{N} \models Con_{PA}$ if and only if PA is a consistent set of axioms. For if PA is not consistent, then PA can prove anything, including $\perp$. If PA is consistent, since we know that there is a proof in PA of $\neg\ \perp$, there must not be a proof of $\perp$.

Theorem 6.6.3 (Gödel's Second Incompleteness Theorem). *If Peano Arithmetic is consistent, then* $PA \nvdash Con_{PA}$.

Proof. Let θ be, as usual, the statement generated by the Self-Reference Lemma, but this time we apply the lemma to the formula $\neg Thm_{PA}(v_1)$. So θ is such that

$$PA \vdash \Big[\theta \leftrightarrow \neg Thm_{PA}\Big(\overline{\ulcorner \theta \urcorner}\Big)\Big]. \tag{6.4}$$

We know that $PA \nvdash \theta$, as PA is a recursive consistent extension of N, all of whose axioms are true in $\mathfrak{N}$ (Proposition 6.4.1). We will show that

$$PA \vdash \big(\theta \leftrightarrow Con_{PA}\big).$$

If it was the case that $PA \vdash Con_{PA}$, then we would have $PA \vdash \theta$, a contradiction. Thus $PA \nvdash Con_{PA}$. [For this proof, all we really need is that $PA \vdash \big(Con_{PA} \rightarrow \theta\big)$, but the other direction is used in the Exercises.]

So all that is left is to show that $PA \vdash \big(\theta \leftrightarrow Con_{PA}\big)$.

For the forward direction, since $\perp$ is the denial of a tautology, we know that

$$PA \vdash (\perp \rightarrow \theta),$$

so by the first derivability condition (D1) we know that

$$PA \vdash Thm_{PA}\big(\overline{\ulcorner \perp \rightarrow \theta \urcorner}\big).$$

This implies, via (D3), that

$$PA \vdash Thm_{PA}(\overline{\ulcorner \perp \urcorner}) \rightarrow Thm_{PA}(\overline{\ulcorner \theta \urcorner}),$$

which is equivalent to

$$PA \vdash \neg Thm_{PA}(\overline{\ulcorner \theta \urcorner}) \rightarrow \neg Thm_{PA}(\overline{\ulcorner \perp \urcorner}). \tag{6.5}$$

Now, if we combine (6.4) and (6.5), we see that

$$PA \vdash \theta \rightarrow \neg Thm_{PA}(\overline{\ulcorner \perp \urcorner}),$$

which is equivalent to

$$PA \vdash \theta \rightarrow Con_{PA},$$

which is half of what we need to prove.

For the converse, notice that from derivability condition (D2),

$$PA \vdash Thm_{PA}(\overline{\ulcorner \theta \urcorner}) \to Thm_{PA}\left(\overline{\ulcorner Thm_{PA}(\overline{\ulcorner \theta \urcorner})\urcorner}\right). \tag{6.6}$$

Since we also know that $PA \vdash \theta \leftrightarrow \neg Thm_{PA}(\overline{\ulcorner \theta \urcorner})$, the sentence

$$Thm_{PA}\left(\overline{\ulcorner Thm_{PA}(\overline{\ulcorner \theta \urcorner}) \to \neg\theta \urcorner}\right)$$

is a true Σ-sentence. Since N suffices to prove true Σ-sentences, certainly

$$PA \vdash Thm_{PA}\left(\overline{\ulcorner Thm_{PA}(\overline{\ulcorner \theta \urcorner}) \to \neg\theta \urcorner}\right). \tag{6.7}$$

Now, if we take (6.7) and derivability condition (D3), we have

$$PA \vdash Thm_{PA}\left(\overline{\ulcorner Thm_{PA}(\overline{\ulcorner \theta \urcorner})\urcorner}\right) \to Thm_{PA}(\overline{\ulcorner \neg\theta \urcorner}). \tag{6.8}$$

If we combine (6.6) and (6.8), we see that

$$PA \vdash Thm_{PA}(\overline{\ulcorner \theta \urcorner}) \to Thm_{PA}(\overline{\ulcorner \neg\theta \urcorner}). \tag{6.9}$$

Now, since we know that the sentence $\theta \to \big[(\neg\theta) \to \bot\big]$ is a tautology, the statement

$$Thm_{PA}\left(\overline{\ulcorner \theta \to \big[(\neg\theta) \to \bot\big] \urcorner}\right)$$

is a true Σ-sentence, so

$$PA \vdash Thm_{PA}\left(\overline{\ulcorner \theta \to \big[(\neg\theta) \to \bot\big] \urcorner}\right). \tag{6.10}$$

Once again, using the derivability condition (D3) twice on (6.10), we find that

$$PA \vdash Thm_{PA}(\overline{\ulcorner \theta \urcorner}) \to \big[Thm_{PA}(\overline{\ulcorner \neg\theta \urcorner}) \to Thm_{PA}(\overline{\ulcorner \bot \urcorner})\big],$$

and if we combine that with (6.9), we see that

$$PA \vdash Thm_{PA}(\overline{\ulcorner \theta \urcorner}) \to Thm_{PA}(\overline{\ulcorner \bot \urcorner}),$$

which is equivalent to

$$PA \vdash \neg Thm_{PA}(\overline{\ulcorner \bot \urcorner}) \to \neg Thm_{PA}(\overline{\ulcorner \theta \urcorner}),$$

which, when we translate the antecedent and use the definition of θ on the consequent, combines with (D3) to give us

$$PA \vdash Con_{PA} \to \theta,$$

which is what we needed to prove. □

The proof of the Second Incompleteness Theorem is technical, but the result is fabulous: If Peano Arithmetic is consistent, it cannot prove its own consistency. You will not be surprised to find out that the same result holds for any consistent set of axioms extending Peano Arithmetic that can be represented by a Σ-formula.

> *Chaff:* Time for a bit of technical stuff that is pretty neat. To be precise, the way in which we code up the axioms of Peano Arithmetic is important in the statement of the Second Incompleteness Theorem. The set AxiomOfPA is representable, and thus there is a formula $\phi(x)$ that represents that set. In fact, the formula $\phi(x)$ can be taken to be a Δ-formula, and if you use *that* ϕ, then Gödel's result is as we have stated it. But there are other formulas that you could use to represent the set of axioms of Peano Arithmetic, and Solomon Feferman proved in 1960 that it is possible to express consistency using one of these other formulas in such a way that $PA \vdash Con_{PA}$ [Feferman 60]. The moral is: You have to be very precise when dealing with consistency statements.

The example of Con_{PA} as a true and yet unprovable statement stood for 45 years as the most natural sentence of that type. In 1977, however, Jeff Paris and Leo Harrington discovered a generalization of Ramsey's Theorem in combinatorics that is not provable in Peano Arithmetic. Since that time a small cottage industry has developed that produces relatively natural statements that are not provable in one theory or another.

We end this section with a couple of interesting corollaries of the Second Incompleteness Theorem. The first corollary is pretty strange. Suppose for a second that Peano Arithmetic is consistent (you believe that it is, right?). Then we can, if we like, assume that PA is *inconsistent*, and we will still have a consistent set of axioms!

Corollary 6.6.4. *If PA is consistent, the set of axioms*

$$PA \cup \{\neg Con_{PA}\}$$

is consistent.

Proof. This is immediate. We know that $PA \nvdash Con_{PA}$. So by Exercise 4 in Section 2.7.1, $PA \cup \{\neg Con_{PA}\}$ is consistent. □

To appreciate the next corollary, remember our development of the First Incompleteness Theorem, in the setting of Peano Arithmetic. The sentence θ that we construct from the Self-Reference Lemma "says" that it (θ) is not provable from PA. Then the Incompleteness Theorem says that θ is right: θ is unprovable from PA, so θ is correct in what it asserts.

Now, suppose that we use the Self-Reference Lemma to construct a different sentence, α, such that α claims that it *is* provable from PA. Is α true? False? Löb's Theorem provides the answer.

Corollary 6.6.5 (Löb's Theorem). *Suppose α is such that*

$$PA \vdash Thm_{PA}(\overline{\ulcorner \alpha \urcorner}) \to \alpha.$$

Then $PA \vdash \alpha$.

Proof. This proof hinges on the fact that PA is sufficiently strong to prove the equivalence

$$\neg Thm_{PA}(\overline{\ulcorner \alpha \urcorner}) \leftrightarrow Con_{PA \cup \{\neg\alpha\}}.$$

Granting this, since by assumption $PA \vdash \big(\neg\alpha \to \neg Thm_{PA}(\overline{\ulcorner \alpha \urcorner})\big)$, we know that

$$PA \cup \{\neg\alpha\} \vdash \neg Thm_{PA}(\overline{\ulcorner \alpha \urcorner}),$$

so by our equivalence,

$$PA \cup \{\neg\alpha\} \vdash Con_{PA \cup \{\neg\alpha\}}.$$

But then by the Second Incompleteness Theorem, $PA \cup \{\neg\alpha\}$ must be inconsistent, and therefore $PA \vdash \alpha$, as needed. □

Löb's Theorem tells us that the sentence that states, "I am provable in Peano Arithmetic" is true (and therefore provable)!

6.6.1 Exercises

1. Explain how the induction schema of Peano Arithmetic as given in Definition 6.6.1 differs from the full principle of mathematical induction:

$$(\forall A \subseteq \mathbb{N})\Big[\big(0 \in A \wedge (\forall x)(x \in A \to Sx \in A)\big) \to A = \mathbb{N}\Big].$$

   [*Suggestion:* You might look at the comment on page 100.]

2. Suppose θ and η are two sentences that assert their own unprovability (from PA). So

$$PA \vdash \Big[\theta \leftrightarrow \neg Thm_{PA}\Big(\overline{\ulcorner \theta \urcorner}\Big)\Big]$$

   and

$$PA \vdash \Big[\eta \leftrightarrow \neg Thm_{PA}\Big(\overline{\ulcorner \eta \urcorner}\Big)\Big].$$

   Prove that $PA \vdash \theta \leftrightarrow \eta$.

3. Let ρ be the Rosser sentence (in the setting of PA). So

$$PA \vdash \Big[\rho \leftrightarrow (\forall x)\big[Deduction_{PA}(x, \overline{\ulcorner \rho \urcorner}) \to (\exists y < x)(Deduction_{PA}(y, \overline{2^{\overline{2}}\overline{3}^{\overline{\ulcorner \rho \urcorner + 1}}}))\big]\Big].$$

Show that

$$PA \vdash \left[Con_{PA} \rightarrow \left[\left(\neg Thm_{PA}(\overline{\ulcorner \rho \urcorner}) \right) \wedge \left(\neg Thm_{PA}(\overline{\ulcorner \neg \rho \urcorner}) \right) \right] \right].$$

Use this and Gödel's Second Incompleteness Theorem to show

$$PA \vdash \left[Con_{PA} \rightarrow \rho \right]$$

and

$$PA \nvdash \left[\rho \rightarrow Con_{PA} \right].$$

So since Gödel's θ is such that $PA \vdash [\theta \leftrightarrow Con_{PA}]$, you have shown that θ and ρ are not provably equivalent in PA.

6.7 Summing Up, Looking Ahead

This chapter is the capstone of this approach to incompleteness. We have stated and proved (with only a small bit of handwaving) the two incompleteness theorems. We are sure that at this point you have a strong understanding of the theorems as well as an appreciation for the delicate arguments that are used in their proofs. These theorems have had a profound impact on the philosophical understanding of the subject of mathematics, and they involve some wonderful mathematics in and of themselves.

The chapter ahead returns to the subject of incompleteness, but attacks the problem in a different way. Rather than focusing on formulas and deductions, the area of computability theory looks at functions and computations. By answering another of Hilbert's Problems, computability theory gives us a new outlook on incompleteness. Fascinating material in and of itself, computability is one of the parts of logic that is closest in feel to computer science. If you want to understand computers, or if you want to gain a deep understanding of one of the ways to analyze just what it means to perform a computation, the next chapter is for you! Have fun!

Chapter 7

Computability Theory

7.1 The Origin of Computability Theory

You are almost certainly familiar with a number of non-trivial algorithms. For example, you know how to find the roots of a quadratic polynomial. Perhaps you remember from a linear algebra course how to row-reduce a matrix. Maybe you even remember how to find the greatest common divisor of two positive integers. Algorithms are omnipresent in mathematics and have been ever since mathematics began. Still, a systematic mathematical theory of algorithms was not developed until the 1930s. This theory used to be called *recursion theory*. Today we use the far more adequate name *computability theory*. Computability theory is regarded as a branch of mathematical logic and has an interesting history.

Can an algorithm decide if a first-order formula is valid? Hilbert posed this question in 1928, and the problem became known as the Entscheidungsproblem (German for "decision problem"). If the answer to the Entscheidungsproblem were to be yes, then it is obvious how the result could established. We would write up an algorithm deciding whether or not a given first-order formula was valid, and then we would argue that this algorithm works correctly. Maybe we could not define what an algorithm *is*, but we would not worry too much about that. Certainly, mathematicians interested in the Entscheidungsproblem would recognize an algorithm when they saw one.

The answer to the Entscheidungsproblem turned out to be no. This was proved independently by Alonzo Church (1935) and Alan M. Turing (1936). Their arguments required a formal mathematical model of the informal notion of an algorithm. Such a model imposes mathematical control over the class of all algorithms and makes it possible to prove that there is no algorithm that decides if a first-order formula is valid.

There are many similar situations in mathematics. If we want to argue that it is possible to carry out a certain geometrical construction by a ruler

and a compass, what can we do? Let us say that we want to demonstrate that we can bisect an angle. Well, we can describe the construction: put the spike of the compass in this point, put a mark on this line, and so on. Then, we explain – by referring to some basic geometrical facts that everyone is likely to accept – why this construction bisects an angle. Such an argument should be sufficient to convince an interested layman. He is not likely to ask for a mathematical model of the physical objects he knows as a ruler and a compass. In contrast, if we want to convince somebody that it is impossible to carry out a geometrical construction by a ruler and a compass, what can we do? Now we are in a different situation. Think a little bit and you will realize that a convincing argument showing it is impossible to trisect an angle by a ruler and a compass requires a mathematical model of the ruler and the compass.

In the early twentieth century there were, in addition to the Entscheidungsproblem, several other open problems that called for a better mathematical understanding of algorithms, computability, and effective calculations; for example, Dehn's word problem and Hilbert's 10th problem. Several formalisms that provided mathematical models of algorithms and computations emerged in the 1930s: Herbrand-Gödel equations, Church's λ-calculus, Kleene recursion, and others. It was obvious that any function computable according to any of these formal systems also was computable in the intuitive sense. The converse question was much more interesting. Although it was not obvious if any of the these formalisms captured the idea of a function being computable in the intuitive sense, experience indicated that this might be the case. There were no examples of functions being computable in the intuitive sense, but not in the formal sense. But why should not such functions exist? Mere inability to produce a counterexample did not imply that a counterexample did not exist.

The situation was clarified when Turing came along with his formal model of computation – the theoretical devices we today know as Turing machines. Such machines model how a human being carries out algorithms by writing and erasing symbols in a spreadsheet. Turing's definition of a computable function was based on an analysis of what human beings actually do when they compute. None of the other definitions of computability were founded on such an analysis. Thus, Turing provided what Gödel, Church, Kleene, and others, could not provide: An argument that justified – or at least made it likely – that every function computable in the intuitive sense also was computable in the formal sense. *Turing's Thesis* states that any function computable in the intuitive sense is computable by a Turing machine. *Church's Thesis* is a similar assertion. Normally we do not bother to distinguish between these two theses. That is why we often talk about the *Church-Turing Thesis*.

Within a short period of time all the above-mentioned formal systems were proved to be equivalent. No matter which of these formalisms we choose to work with, we will end up with the same class of computable

functions. We have not gained any further intuition since the 1930s that would indicate that this class of functions should be enlarged. We have not encountered any functions outside this class that it would be reasonable to call computable. Turing analyzed humans computing with pen and paper. Others, e.g., Robin Gandy, have analyzed idealized, discrete, deterministic machines following the laws of classical mechanics. The conclusion is that such machines cannot compute any functions that cannot be computed by humans. The same goes for machines based on quantum mechanics.

Our introduction to computability theory is based on the work of the American mathematician Stephen Kleene. Kleene recursion deals with functions from the natural numbers into the natural numbers. If our introduction were based on Turing machines, the λ-calculus, or a modern commercial programming language, our results and theorems would still be the same. The conclusion of the discussion above is that our results have a character of absoluteness. If it follows from our formal – and maybe very technical – definitions that a function is not computable, then we should conclude that this function is not computable in a certain absolute sense. And when the same formal definitions entail that a problem – like the Entscheidungsproblem – is undecidable, we should conclude that the problem is undecidable in a certain absolute sense.

7.2 The Basics

We will use Kleene recursion – often also called μ-recursion – to define the computable functions. Why do we not use Turing Machines for this purpose? We have argued that Turing machines played a unique and pivotal role in the history of computability theory. So why do we not use them? Besides, there are quite a few other formalisms that in principle we could use: Markov algorithms, the untyped lambda-calculus, the typed lambda-calculus extended with basic functions and a fixed point operator (PCF), register machines, stack machines, several (idealized) programming languages, string rewriting systems, term rewriting systems, graph manipulating machines, abstract state machines, tiling systems, cellular automata, ... So, why do we pick Kleene recursion?

Each of the above-mentioned formal systems provides a Turing-complete model of computation, that is, a model of computation that induces the same class of computable functions as the Turing machines. Which model to choose depends on which aspects of computations you are interested in. A model that will be suitable for studying some aspects will be unsuitable for studying other aspects. This is a textbook on mathematical logic. Our goal is to give an introduction to basic computability theory, and then use this theory to prove some classical theorems of logic. We use Kleene recursion for our presentation since we can, to a large extent, rely on notions and notation familiar to any student of mathematics.

Kleene recursion provides us with a compact and elegant inductive definition of the class of computable functions (Definition 7.2.1). This definition gives our presentation and our proofs a nice structure, moreover, the definition yields the primitive recursive functions for free. The primitive recursive functions make up an interesting class of total computable functions with which you should have at least a passing acquaintance. It will also be convenient to us that the Kleene recursive functions are functions on the natural numbers. That makes it easy to bridge the gap between our model of computation and the standard $\mathcal{L}_{NT}$-structure $\mathfrak{N}$.

Let us start to build the collection of Kleene-recursive, or computable, functions. We will first give the general idea before formalizing our definition in Definition 7.2.1.

Our computable functions will be functions from the natural numbers into the natural numbers. We start with some initial functions, very simple functions that you would agree are all computable in the intuitive sense. Here we go:

For each $i, n \in \mathbb{N}$ such that $1 \leq i \leq n$, the projection function $\mathcal{I}_i^n : \mathbb{N}^n \to \mathbb{N}$ is an initial function. The function $\mathcal{I}_i^n$ is defined by $\mathcal{I}_i^n(x_1, \ldots, x_n) = x_i$, e.g., $\mathcal{I}_2^4(17, 3, 9, 7) = 3$ and $\mathcal{I}_1^1(17) = 17$. The only initial functions apart from the projection functions are the successor function $\mathcal{S} : \mathbb{N} \to \mathbb{N}$ and the function $\mathcal{O}$. The function $\mathcal{S}$ applied to the natural number x yields the natural number $x + 1$, and $\mathcal{O}$ is the function that takes no arguments and yields the natural number 0. Some would say that $\mathcal{O}$ is a constant and not a function, but to us it will be convenient to view $\mathcal{O}$ as a function of arity 0.

We can define new computable functions from given ones by setting up equations, but we are not allowed to use arbitrary equations. The equations are required to be in certain forms. These forms are given by what we will call *definition schemes*.

Let m and n be any natural numbers. The definition scheme

$$f(x_1, \ldots, x_n) = h(g_1(x_1, \ldots, x_n), \ldots, g_m(x_1, \ldots, x_n)) \qquad \text{(Composition)}$$

shows how we can define a function f of arity n from given functions $g_1, \ldots, g_m$ of arity n and h of arity m. This scheme is called *composition*. If f is defined by this scheme and we know how to compute the functions $g_1, \ldots, g_m$ and h, then we also know how to compute the function f. To determine the value of $f(x_1, \ldots, x_n)$, we compute the value v_1 of $g_1(x_1, \ldots, x_n)$, the value v_2 of $g_2(x_1, \ldots, x_n)$, and so on, until we have computed the value v_m of $g_m(x_1, \ldots, x_n)$, and finally we find the value of $f(x_1, \ldots, x_n)$ by computing the value of $h(v_1, \ldots, v_m)$.

The formal scheme for composition given above is rigid and does not allow us to compose functions the way we are used to, e.g., the composition

$$f(x_1, x_2, x_3) = h(g_1(x_2, x_1, x_1), g_2(x_3)) \qquad (*)$$

does not satisfy the scheme. The scheme

$$f(x_1, \ldots, x_n) = h(g_1(x_1, \ldots, x_n), \ldots, g_m(x_1, \ldots, x_n))$$

requires the functions $g_1, \ldots, g_m$ to be of the same arity and to be fed with the same list of arguments. However, when projection functions are available, natural compositions like (*) can be achieved by repeated applications of the formal scheme. The following three compositions

(1) $j_1(x_1, x_2, x_3) = g_1(\mathcal{I}^3_2(x_1, x_2, x_3), \mathcal{I}^3_1(x_1, x_2, x_3), \mathcal{I}^3_1(x_1, x_2, x_3))$

(2) $j_2(x_1, x_2, x_3) = g_2(\mathcal{I}^3_3(x_1, x_2, x_3))$

(3) $f(x_1, x_2, x_3) = h(j_1(x_1, x_2, x_3), j_2(x_1, x_2, x_3))$

stick to the formal scheme, and

$$\begin{aligned}
&f(x_1,x_2, x_3)\\
&\quad \overset{(3)}{=} h(j_1(x_1, x_2, x_3), j_2(x_1, x_2, x_3))\\
&\quad \overset{(1)}{=} h(g_1(\mathcal{I}^3_2(x_1, x_2, x_3), \mathcal{I}^3_1(x_1, x_2, x_3), \mathcal{I}^3_1(x_1, x_2, x_3)), j_2(x_1, x_2, x_3))\\
&\quad = h(g_1(x_2, x_1, x_1), j_2(x_1, x_2, x_3))\\
&\quad \overset{(2)}{=} h(g_1(x_2, x_1, x_1), g_2(\mathcal{I}^3_3(x_1, x_2, x_3)))\\
&\quad = h(g_1(x_2, x_1, x_1), g_2(x_3)) \ .
\end{aligned}$$

Thus, we have reduced (*) to a number of compositions, each in accordance with the formal scheme. Any natural composition of functions can be reduced to compositions satisfying the formal scheme.

The second way that we can create new computable functions from old ones is called *primitive recursion*. Let n be any natural number, and let $\underset{\sim}{x} = x_1, \ldots, x_n$. The definition scheme

$$\begin{aligned}
f(\underset{\sim}{x}, 0) &= g(\underset{\sim}{x})\\
f(\underset{\sim}{x}, y+1) &= h(\underset{\sim}{x}, y, f(\underset{\sim}{x}, y)) \qquad \text{(Primitive Recursion)}
\end{aligned}$$

shows how we can define a function f of arity $n+1$ from a function g of arity n and a function h of arity $n+2$. We can compute the function f defined by this scheme if we know how to compute g and h. To determine the value v_0 of $f(\underset{\sim}{x}, 0)$, we compute the value of $g(\underset{\sim}{x})$. To determine the value v_1 of $f(\underset{\sim}{x}, 1)$, we compute the value of $h(\underset{\sim}{x}, 0, v_0)$. To determine the value v_2 of $f(\underset{\sim}{x}, 2)$, we compute the value of $h(\underset{\sim}{x}, 1, v_1)$. And thus we proceed until we have found the value of $f(\underset{\sim}{x}, y)$ for the desired y. Note that

$$f(\underset{\sim}{x}, y) \quad = \quad h(\underset{\sim}{x}, y-1, h(\underset{\sim}{x}, y-2, \ldots h(\underset{\sim}{x}, 0, g(\underset{\sim}{x})) \ldots))$$

and that we have a procedure for computing the function f by first executing the procedure for computing g once, and then, executing the procedure for computing h many times in a row. To compute f with the arguments $\underset{\sim}{x}, y$, we need to execute the procedure for computing h exactly y times.

If we can define a function f from the initial functions ($\mathcal{O}$, $\mathcal{S}$, and projections) by using the scheme of composition and/or the scheme of primitive recursion, we will say that f is defined *primitive recursively* and that we have have a *primitive recursive definition* of f. The *primitive recursive functions* are those functions that have primitive recursive definitions.

Let us study an example: First, we define the function h from the initial functions $\mathcal{S}$ and $\mathcal{I}_3^3$ by the scheme of composition:

$$h(x,y,z) \;=\; \mathcal{S}(\mathcal{I}_3^3(x,y,z)) \qquad \text{(i)}$$

Next, we define the function f from h and the initial function $\mathcal{I}_1^1$ by the scheme of primitive recursion:

$$\begin{aligned} f(x,0) &= \mathcal{I}_1^1(x) && \text{(ii)}\\ f(x,y+1) &= h(x,y,f(x,y)) && \text{(iii)} \end{aligned}$$

Now, we have a perfectly nice primitive recursive definition of a function $f : \mathbb{N}^2 \to \mathbb{N}$. We claim that $f(x,y) = x + y$.

Let us try to find a more readable definition of f. Observe that $h(x,y,z) = \mathcal{S}(\mathcal{I}_3^3(x,y,z)) = \mathcal{S}(z)$ and $\mathcal{I}_1^1(x) = x$. Thus, in place of (i), (ii) and (iii), we may define f by

$$\begin{aligned} f(x,0) &= x\\ f(x,y+1) &= \mathcal{S}(f(x,y)) \end{aligned}$$

Now, these two equations are not in accordance with our formal schemes, but it is fairly easy to see that they can be reduced to equations satisfying the formal schemes. Moreover, these equations make it easier to realize that

$$f(x,y) \;=\; \underbrace{\mathcal{S}(\mathcal{S}(\ldots\mathcal{S}(}_{y \text{ times}}x)\ldots)) \;=\; x+y\,.$$

Now, if we use the symbol $+$ in place of the symbol f and write our two equations in infix notation, we get

$$\begin{aligned} x+0 &= x\\ x+(y+1) &= \mathcal{S}(x+y) \end{aligned}$$

These are two transparent equations that define the addition function. These two equations make it easy to realize that addition on natural numbers is a function that can be defined from the initial functions by the scheme of composition and the scheme of primitive recursion.

When we need to argue that a function is primitive recursive, we usually will not write up equations that strictly follow the formal definition schemes in all their gory detail. This would be an insurmountable task, and besides, such an effort would be of no use as our definitions would turn out to be totally unreadable. Hence, when we need to give a primitive recursive definition of a function, we will put up some informal – but hopefully comprehensible and informative – equations that define the function, and then we will trust that the reader realizes that these equations can be reduced to equations in accordance with the formal schemes. We will, for example, very rarely use our our formal composition scheme. We will compose functions freely, that is, the way we are used to from ordinary mathematics. When projection functions are available, such free compositions can always be reduced to definitions that satisfy our formal composition scheme.

All the initial functions are total functions, functions whose domain is $\mathbb{N}$. Furthermore, when we define a function by the scheme of composition – or by the scheme of primitive recursion – over total functions, we get another total function. Hence, all primitive recursive functions are total functions. We will now introduce a definition scheme that allows us to define partial functions.

Let g be a function of arity $n+1$. Then, $(\mu i)[g(x_1, \ldots, x_n, i)]$ denote the least natural number i such that

- for any $j < i$, the value of $g(x_1, \ldots, x_n, j)$ is a natural number different from 0
- the value of $g(x_1, \ldots, x_n, i)$ is the natural number 0.

We can view $(\mu i)[\ldots]$ as an operator on functions. If we apply the operator to a function of $n+1$ variables, we get a function of n variables. The operator is sometimes called the *μ-operator*, sometimes called the *least number operator*, and sometimes called the *minimalization operator*. The definition scheme

$$f(x_1, \ldots, x_n) \quad = \quad (\mu i)[g(x_1, \ldots, x_n, i)] \qquad \text{(Minimalization)}$$

is called *minimalization*. When this scheme is available, we can define a function f of arity n by applying the μ-operator to a function g of arity $n+1$.

The result of applying the μ-operator to a total function will not necessarily be a total function. Thus, by introducing minimalization, we also introduce partial functions. For example, let $f(x) = (\mu i)[g(x, i)]$, where $g(u, v) = 0$ if $u = v^2$, and $g(u, v) = 1$ if $u \neq v^2$. Then, g is a total function, but f is not. If x is not a perfect square, then the value of $f(x)$ is undefined. (If x is a square, the value of $f(x)$ is defined and equals $\sqrt{x}$.)

If f is defined by minimalization over g and we know how to compute g, then we can compute $f(\underset{\sim}{x})$ by

- first attempt to compute the value v_0 of $g(\underset{\sim}{x}, 0)$
- then attempt to compute the value v_1 of $g(\underset{\sim}{x}, 1)$
- then attempt to compute the value v_2 of $g(\underset{\sim}{x}, 2)$
- ... and so on ...

until we find an i such that the value v_i of $g(\underset{\sim}{x}, i)$ is 0. Then, i will be the value of $f(\underset{\sim}{x})$. This procedure for computing $f(\underset{\sim}{x})$ may not terminate. One reason might be that there is no i such that $g(\underset{\sim}{x}, i)$ equals 0. If so, we will go on forever, computing $g(\underset{\sim}{x}, i)$ for larger and larger values of i. Another reason might be that we encounter an i such that our attempt to compute $g(\underset{\sim}{x}, i)$ does not terminate. Then, obviously, our computation of $f(\underset{\sim}{x})$ will not terminate either. Anyway, no matter what the reason might be, if the computation does not terminate, the value of $f(\underset{\sim}{x})$ is undefined.

Partial functions are important in computability theory. So are computations of functions, that is, algorithms that take the arguments of a function as inputs and yield the value of the function in those arguments as output. Two very different algorithms may compute the same function. When we claim that an algorithm computes a partial function f of arity n, we claim that

- the algorithm does not terminate on the inputs $x_1, \ldots, x_n \in \mathbb{N}$ if $f(x_1, \ldots, x_n)$ is undefined
- the algorithm terminates on the inputs $x_1, \ldots, x_n \in \mathbb{N}$ and yields the output $y \in \mathbb{N}$ if $f(x_1, \ldots, x_n)$ is defined and equals y.

Observe that our algorithms require natural numbers as inputs, 'undefined' is not a value that we might pass on as input to an algorithm. If f is a computable function, then $f(a_1, \ldots, a_n)$ will not be defined if some of the arguments $a_1, \ldots, a_n$ are not defined. For example, let $h(x) = \mathcal{I}_1^2(17, g(x))$. Then, $h(x)$ is undefined if $g(x)$ is undefined. If $g(x)$ is defined, then $h(x)$ is defined and equals 17.

Every computable function can be defined from the initial functions by repeated applications of the scheme of composition, the scheme of primitive recursion and the scheme of minimalization. Our fundamental computational objects are the functions. Computable sets, relations and predicates are defined in terms of computable functions.

After this lengthy bit of motivation, we are now ready to give our official definition of the computable functions and the computable sets.

Definition 7.2.1. We define the set of *computable functions* inductively by the following six clauses.

(1) the successor function $\mathcal{S}$ is a computable function

(2) the projection function $\mathcal{I}_i^n$ is a computable function (for all $i, n \in \mathbb{N}$ such that $1 \leq i \leq n$)

(3) the zero function $\mathcal{O}$ is a computable function

(4) f is a computable function if there are computable functions $g_1, \ldots, g_m$, and h such that

$$f(x_1, \ldots, x_n) = h(g_1(x_1, \ldots, x_n), \ldots, g_m(x_1, \ldots, x_n)) \qquad \text{(Composition)}$$

for all $x_1, \ldots, x_n \in \mathbb{N}$

(5) f is a computable function if there are computable functions g and h such that

$$\begin{aligned} f(x_1, \ldots, x_n, 0) &= g(x_1, \ldots, x_n) \\ f(x_1, \ldots, x_n, y+1) &= h(x_1, \ldots, x_n, y, f(x_1, \ldots, x_n, y)) \end{aligned} \qquad \text{(Primitive Recursion)}$$

for all $x_1, \ldots, x_n, y \in \mathbb{N}$

(6) f is a computable function if there is a computable function g such that

$$f(x_1, \ldots, x_n) = (\mu i)[g(x_1, \ldots, x_n, i)] \qquad \text{(Minimalization)}$$

for all $x_1, \ldots, x_n \in \mathbb{N}$.

Chaff: Notice, in the above, that the domains of the already-known-to-be computable functions g and h define the domain of the function f. So, for example, if $f(x) = h(g(x))$ and $17 \notin \text{dom}(g)$, or $g(17) \notin \text{dom}(h)$, then $17 \notin \text{dom}(f)$.

Definition 7.2.2. The set of *primitive recursive functions* is defined inductively using clauses (1) through (5) from the definition of the set of computable functions, but without Clause (6).

Definition 7.2.3. Let A be a set of natural numbers or tuples of natural numbers, that is, $A \subseteq \mathbb{N}^n$ for some $n \geq 1$. We define the *characteristic function χ_A for A*, by

$$\chi_A(x_1, \ldots, x_n) = \begin{cases} 0 & \text{if } (x_1, \ldots, x_n) \in A \\ 1 & \text{otherwise} \end{cases}$$

Definition 7.2.4. A *set is computable* when the characteristic function for the set is computable. A *set is primitive recursive* when the characteristic function for the set is primitive recursive.

Relations and predicates are treated as sets: formally, an n-ary relation is nothing but a set of n-tuples, and a predicate is just another name for a relation. For example, the characteristic function for the standard strict ordering relation $<$ on the natural numbers is the function $\chi_<$ given by

$$\chi_<(x,y) = \begin{cases} 0 & \text{if } x < y \\ 1 & \text{otherwise.} \end{cases}$$

When R is a relation, we will write $(\mu i)[R(\underset{\sim}{x}, i)]$ in place of $(\mu i)[\chi_R(\underset{\sim}{x}, i)]$. Thus, $(\mu i)[R(\underset{\sim}{x}, i)]$ denotes the least i such that the relation $R(\underset{\sim}{x}, i)$ holds. For example, $(\mu i)[x < i]$ denotes the least i strictly larger than x, that is, the successor of x.

7.3 Primitive Recursion

In this section we prove a number of very basic results. We shall see that many familiar functions can be defined primitive recursively, that is, without applying the scheme of minimalization. Indeed, you have to work hard to come up with a total computable function that is not primitive recursive (see Exercise 9).

In order to provide some intuition, we will discuss how multiplication, exponentiation, and a few other well-known functions can be defined primitive recursively. We have already seen that addition can be defined by

$$\begin{aligned} x + 0 &= x && \text{(i)} \\ x + (y+1) &= \mathcal{S}(x+y) && \text{(ii)} \end{aligned}$$

and that these two equations can be reduced to equations satisfying our formal definition schemes. By (i) and (ii), we have

$$4{+}3 \overset{\text{(ii)}}{=} \mathcal{S}(4{+}2) \overset{\text{(ii)}}{=} \mathcal{S}(\mathcal{S}(4{+}1)) \overset{\text{(ii)}}{=} \mathcal{S}(\mathcal{S}(\mathcal{S}(4{+}0))) \overset{\text{(i)}}{=} \mathcal{S}(\mathcal{S}(\mathcal{S}(4))) = 7.$$

Thus, $4 + 3$ equals the result of iterating the successor function 3 times in a row on the argument 4. In general, we have

$$a + b = \underbrace{\mathcal{S}(\mathcal{S}(\ldots\mathcal{S}(}_{b \text{ times}} a)\ldots)).$$

In the same way that we can define addition primitive recursively by iterating the successor function, we can define multiplication primitive recursively by iterating the addition function. When we put up the equations

$$\begin{aligned} x \cdot 0 &= 0 && \text{(iii)} \\ x \cdot (y+1) &= x + (x \cdot y) && \text{(iv)} \end{aligned}$$

we have

$$4 \cdot 3 \overset{\text{(iv)}}{=} 4 + (4 \cdot 2) \overset{\text{(iv)}}{=} 4 + (4 + (4 \cdot 1)) \overset{\text{(iv)}}{=} 4 + (4 + (4 + (4 \cdot 0))) \overset{\text{(iii)}}{=} 4 + (4 + (4 + 0)) = 12 .$$

In general, the equations yield

$$a \cdot b = \underbrace{a + a + \ldots a +}_{b \text{ times}} 0 .$$

It is an exercise for the reader to verify that (iii) and (iv) can be reduced to equations satisfying our formal definition schemes (Exercise 5).

Once we have defined multiplication, we can go on and define exponentiation as iterated multiplication. The equations $x^0 = 1$ and $x^{y+1} = x \cdot (x^y)$ yield

$$a^b = \underbrace{a \cdot a \cdot \ldots \cdot a \cdot}_{b \text{ times}} 1 .$$

We can proceed along this line. Once we have defined exponentiation, we can define super-exponentiation by iterating exponentiation, and then define a function that grows considerably faster than super-exponentiation by iterating super-exponentiation ... and thus we can proceed. Needless to say, primitive recursive functions may grow very very fast. However, such insanely fast-growing functions will not be important to us.

To find primitive recursive definitions of functions that do not grow at all, e.g., functions that decrease or only take the values 0 and 1, we have to start off by defining the predecessor function, that is, the function P given by $P(0) = 0$ and $P(n) = n - 1$ for $n > 0$. The equations

$$\begin{aligned} P(0) &= \mathcal{O} \\ P(y+1) &= \mathcal{I}_1^2(y, P(y)) \end{aligned}$$

satisfy our formal definition scheme (primitive recursion) and define the predecessor function. Once the predecessor is defined, we can proceed and define other non-increasing functions primitive recursively.

It is time to round off our little informal discussion. We will now proceed systematically and prove that a number of familiar functions can be defined primitive recursively.

Lemma 7.3.1. *Let $i, n \in \mathbb{N}$, and let c_i^n be the n-ary constant function given by*

$$c_i^n(x_1, \ldots, x_n) = i .$$

Then, c_i^n is primitive recursive for every $n, i \in \mathbb{N}$.

Proof. We have $c_0^0 = \mathcal{O}$. Thus, c_0^0 is primitive recursive as $\mathcal{O}$ is one of the initial functions. When $k > 0$, we can define define c_0^k by composition. Recall the scheme of composition:

$$f(x_1, \ldots, x_n) = h(g_1(x_1, \ldots, x_n), \ldots, g_m(x_1, \ldots, x_n)) \, .$$

We apply the scheme with 0 for m and k for n, and we define c_0^k by $c_0^k(x_1, \ldots, x_k) = \mathcal{O}$.

Assume, by induction hypothesis, that c_i^k is primitive recursive (for every $k \in \mathbb{N}$). We define c_{i+1}^k by composition of the two primitive recursive functions c_i^k and $\mathcal{S}$, that is, $c_{i+1}^k = \mathcal{S}(c_i^k(x_1, \ldots, x_k))$. Thus, we conclude that c_i^n is primitive recursive for every $n, i \in \mathbb{N}$. □

Lemma 7.3.2. *The functions $+$, $\cdot$ and x^y (standard addition, multiplication, and exponentiation) are primitive recursive.*

Proof. Let c_0^1 and c_1^1 be the constant functions from Lemma 7.3.1. We have

- $x + 0 = x$ and $x + (y + 1) = \mathcal{S}(x + y)$
- $x \cdot 0 = c_0^1(x)$ and $x \cdot (y + 1) = x + (x \cdot y)$
- $x^0 = c_1^1(x)$ and $x^{y+1} = x \cdot x^y$.

Thus, we conclude that $+$, $\cdot$ and x^y are primitive recursive functions (see the discussion above for more details). □

Lemma 7.3.3. *The modified subtraction function $\dot{-}$ is given by*

$$x \dot{-} y = \begin{cases} 0 & \text{if } y > x \\ x - y & \text{(ordinary subtraction) otherwise.} \end{cases}$$

The function $\dot{-}$ is primitive recursive.

Proof. We define the function P by primitive recursion: $P(0) = \mathcal{O}$ and $P(y + 1) = \mathcal{I}_1^2(y, P(y))$. Now, P is the predecessor function, that is, we have $P(0) = 0$, and for $x > 0$, we have $P(x) = x - 1$. We have $x \dot{-} 0 = x$ and $x \dot{-} (y + 1) = P(x \dot{-} y)$. Thus, we conclude that $\dot{-}$ is primitive recursive. □

Lemma 7.3.4. *The primitive recursive relations are closed under the connectives of propositional logic.*

Proof. Let χ_R be the characteristic function for the primitive recursive relation $R(\underset{\sim}{x})$, and let χ_S be the characteristic function for the primitive recursive relation $S(\underset{\sim}{x})$. The primitive recursive function $\chi_R(\underset{\sim}{x}) \cdot \chi_S(\underset{\sim}{y})$ is the characteristic function for the relation $R(\underset{\sim}{x}) \vee S(\underset{\sim}{y})$, and the primitive recursive function $1 \dot{-} \chi_R(\underset{\sim}{x})$ is the characteristic function for the relation $\neg R(\underset{\sim}{x})$. All other propositional connectives can be expressed by $\vee$ and $\neg$. Thus, we conclude that our lemma holds. □

Lemma 7.3.5. *The standard ordering relations $\leq, <$ (on the natural numbers) are primitive recursive. The equality relation $=$ is primitive recursive.*

Proof. The primitive recursive function $1 \dot{-} ((y+1) \dot{-} x)$ is the characteristic function for the relation $x \leq y$. Furthermore, we have

- $x = y \;\Leftrightarrow\; x \leq y \wedge y \leq x$
- $x < y \;\Leftrightarrow\; \neg(y \leq x)$.

Thus, it follows from Lemma 7.3.4 that these relations are primitive recursive. □

Lemma 7.3.6. *The class of primitive recursive functions is closed under bounded sum and bounded product, that is, $\sum_{i \leq y} f(\underset{\sim}{x}, i)$ and $\prod_{i \leq y} f(\underset{\sim}{x}, i)$ are primitive recursive functions if f is a primitive recursive function.*

Proof. We have

$$\sum_{i \leq 0} f(\underset{\sim}{x}, i) \;=\; f(\underset{\sim}{x}, 0) \quad \text{and} \quad \sum_{i \leq y+1} f(\underset{\sim}{x}, i) = f(\underset{\sim}{x}, \mathcal{S}(y)) + \sum_{i \leq y} f(\underset{\sim}{x}, i)\ .$$

Thus, it is easy to see that we can define the function $\sum_{i \leq y} f(\underset{\sim}{x}, i)$ primitive recursively from the function f, and we conclude that the class of primitive recursive function is closed under bounded sum.

It is also easy to see that the class of primitive recursive function is closed under bounded products. It is straightforward to define the product $\prod_{i \leq y} f(\underset{\sim}{x}, i)$ primitive recursively when the function f and the multiplication function are available. □

Lemma 7.3.7. *The class of primitive recursive relations is closed under the bounded first-order quantifiers $(\exists i \leq n)$ and $(\forall i \leq n)$.*

Proof. We have just proved that the class of primitive recursive functions is closed under bounded products. If χ_R is the characteristic function for the relation $R(\underset{\sim}{x}, y)$, then $\prod_{i \leq y} \chi_R(\underset{\sim}{x}, i)$ will be the characteristic function for the relation $(\exists i \leq y)[R(\underset{\sim}{x}, i)]$. Hence, the class of primitive recursive relations is closed under bounded existential quantification. Furthermore, we know that class is closed under the propositional operators, and we know that $(\forall i \leq y)[R(\underset{\sim}{x}, i)]$ holds if and only if $\neg(\exists i \leq y)[\neg R(\underset{\sim}{x}, i)]$ holds. Thus, we can conclude that the class also is closed under bounded universal quantification. □

Definition 7.3.8. The definition scheme

$$f(x_1, \ldots, x_n) = \begin{cases} g_1(x_1, \ldots, x_n) & \text{if } h(x_1, \ldots, x_n) = 0 \\ g_2(x_1, \ldots, x_n) & \text{otherwise} \end{cases}$$

shows how a function f is defined from the functions g_1, g_2, and h. This scheme is called *definition by cases.*

The proof of the next lemma is an exercise for the reader (see Exercise 2).

Lemma 7.3.9. *The class of primitive recursive functions is closed under definition by cases.*

Definition 7.3.10. We will use $(\mu i \leq y)[g(\underset{\sim}{x}, i)]$ to denote the least number i such that $i \leq y$ and $g(\underset{\sim}{x}, i) = 0$. If no such i exists, then $(\mu i \leq y)[g(\underset{\sim}{x}, i)]$ denotes the number $y + 1$. The definition scheme

$$f(\underset{\sim}{x}, y) = (\mu i \leq y)[g(\underset{\sim}{x}, i)] = \begin{cases} \text{the least } i \text{ such that } i \leq y \text{ and } g(\underset{\sim}{x}, i) = 0 \\ y + 1 \text{ if such an } i \text{ does not exist} \end{cases}$$

is called *bounded minimalization* and shows how a function f is defined from a function g.

When R is a relation, we will write $(\mu i \leq y)[R(\underset{\sim}{x}, i)]$ in place of $(\mu i \leq y)[\chi_R(\underset{\sim}{x}, i)]$. (Recall that χ_R denotes the characteristic function for the relation R.) Thus, if there exists i less than or equal to n such a relation $R(\underset{\sim}{x}, i)$ holds, then $(\mu i \leq n)[R(\underset{\sim}{x}, i)]$ will yield the least such i.

Let us look at a couple of examples. Let

$$g(x_1, x_2) \;=\; (\mu i \leq x_2)\,[\,x_1 < i \;\wedge\; i < x_2 \;\wedge\; (\exists z \leq x_2)[z \cdot 2 = i]\,]\,.$$

Then, $g(x_1, x_2)$ yields an even number that lies strictly between x_1 and x_2 if such a number exists. If several such numbers exist, the function yields the least one. If no such numbers exist, then $g(x_1, x_2)$ equals $x_2 + 1$.

For a second example, let

$$f(x, y) \;=\; (\mu i \leq x)\,[\,x < y \cdot (i + 1)\,]\,.$$

Then, if $y > 0$, the function $f(x, y)$ will yield the number $\lfloor x/y \rfloor$ (x divided by y rounded down).

Lemma 7.3.11. *The class of primitive recursive functions is closed under bounded minimalization.*

Proof. Let g be a primitive recursive function. We have to prove that there exists a primitive recursive function f such that $f(\underset{\sim}{x}, y) = (\mu i \leq y)[g(\underset{\sim}{x}, i)]$. To improve the readability we will assume that the argument list $\underset{\sim}{x}$ is empty. It is straightforward to generalize our proof to the case when $\underset{\sim}{x}$ is a list of arbitrary length.

Let $\chi(y)$ denote the characteristic function for the relation $(\exists i \leq y)[g(i) = 0]$. The function χ is primitive recursive by the lemmas above. Moreover, we have

$$\chi(z) = \begin{cases} 0 & \text{if there exist } i \text{ such that } i \leq z \text{ and } g(i) = 0 \\ 1 & \text{if } g(i) \neq 0 \text{ for all } i \leq z. \end{cases}$$

Let $f(y) = \sum_{i \leq y} \chi(i)$. Then, f is primitive recursive by the lemmas above. First, assume there is no i such that $i \leq y$ and $g(i) = 0$. Then we have

$$\begin{array}{rcl} f(0) & = & 1 \\ f(1) & = & f(0) + 1 \quad = \quad 2 \\ f(2) & = & f(1) + 1 \quad = \quad 3 \\ & \vdots & \\ f(y) & = & f(y-1) + 1 \quad = \quad y + 1 \end{array}$$

Next, assume that i is the least number such that $i \leq y$ and $g(i) = 0$. Then we have

$$f(y) \quad = \quad \overbrace{1 \ + \ 1 \ + \ \ldots \ + \ 1}^{\text{this sum is } i} \ + \ \overbrace{0 \ + \ 0 \ + \ \ldots \ + \ 0}^{(y+1) - m \text{ zeroes}} .$$

Thus, $f(y) = (\mu i \leq y)[g(i)]$. □

In Chapter 4 we defined the function $p : \mathbb{N} \to \mathbb{N}$. Recall that $p(0) = 1$ and $p(k)$ is the k^{th} prime for $k \geq 1$. Thus $p(0) = 1$, $p(1) = 2$, $p(2) = 3$, and so on.

Lemma 7.3.12. *The function p is primitive recursive.*

Proof. Let $C(x)$ be the predicate given by

$$C(x) \quad \Leftrightarrow \quad \neg(x \geq 2 \ \wedge \ (\forall y \leq x)(\forall z \leq x)[(y+2) \cdot (z+2) \neq x]) .$$

This predicate states that x *is not* a prime. Let χ_C be the characteristic function of C. Now, $\chi_C(x) = 1$ when x is a prime, and $\chi_C(x) = 0$ when x is not a prime. Let $g(y) = \sum_{i \leq y} \chi_C(i)$, and $g(y)$ yields the number of primes less than or equal to y. It is a fact that the n^{th} prime is less than or equal to $2^{(2^n)}$. This entails that

$$p(n) = \begin{cases} 1 & \text{if } n = 0 \\ (\mu i \leq 2^{(2^n)})[g(i) = n] & \text{otherwise} \end{cases}$$

and thus, it follows from the lemmas above that p is a primitive recursive function. □

Lemma 7.3.13. *For $i > 0$, let $\pi_i(m)$ be the function which yields the exponent of the prime $p(i)$ in the prime factorization of m when $m > 1$. Let $\pi_i(0) = \pi_i(1) = 0$ (for $i \in \mathbb{N}$). The function $\pi_i(m)$ is primitive recursive (for any $i > 0$).*

Proof. The exponent of $p(i)$ in the prime factorization of m will be the greatest j such that $p(i)^j$ divides m. Thus,

$$\pi_i(m) = \begin{cases} (\mu j \leq m)\,[\,(\exists x \leq m)[x \cdot p(i)^j = m] \wedge \\ \qquad\qquad (\forall x \leq m)[x \cdot p(i)^{j+1} \neq m]\,] & \text{if } m > 1 \\ 0 & \text{otherwise.} \end{cases}$$

It follows from the lemmas above that $\pi_i(m)$ is a primitive recursive function. □

Note that we have, for example,

$$\pi_1(p(1)^7 \cdot p(2)^3 \cdot p(3)^{47}) = 7 \quad \text{and} \quad \pi_2(p(1)^7 \cdot p(2)^3 \cdot p(3)^{47}) = 3$$
$$\text{and} \quad \pi_3(p(1)^7 \cdot p(2)^3 \cdot p(3)^{47}) = 47.$$

For $i > 3$, we have $\pi_i(p(1)^7 \cdot p(2)^3 \cdot p(3)^{47}) = 0$.

Coding of sequences is important in computability theory. The previous lemmas tell us that we can code and decode sequences of numbers by primitive recursive means. We need the coding conventions introduced in Definitions 4.5.3, 4.5.5, and 4.5.6 and then we can base our coding system on the fact that natural numbers have unique prime factorizations. Referring to the definitions and motivation in Chapter 4, we leave the proofs of the next few lemmas to the reader.

Lemma 7.3.14. *We have primitive recursive functions $(x)_i$ and $|x|$ such that for every sequence $a_1, \ldots, a_n$ of natural numbers there exists a natural number a such that $|a| = n$ and $(a)_i = a_i$ for $i = 1, \ldots, n$.*

We say that a is a *code* for the sequence $(a_1, \ldots, a_n)$ if $|a| = n$ and $(a)_i = a_i$ for $i = 1, \ldots, n$. Some natural numbers will not be a code for a sequence (recall our discussion in Chapter 4).

Lemma 7.3.15. *We have a primitive recursive predicate* code *such that* code(a) *holds iff a is a code for a sequence. Furthermore, we have primitive recursive functions ε (of arity 0), $\langle x \rangle$ and $x^\frown y$ such that*

(1) ε is a code for the empty sequence, that is, $|\varepsilon| = 0$

(2) $\langle a \rangle$ yields the code for the sequence a of length 1, that is, $|\langle a \rangle| = 1$ and $(\langle a \rangle)_1 = a$

(3) if a is a code for the sequence $(a_1, \ldots, a_n)$ and b is a code for the sequence $(b_1, \ldots, b_m)$, then $a^\frown b$ is a code for the sequence

$$(a_1, \ldots, a_n, b_1, \ldots, b_m).$$

We will say that $(x)_i$ is a *decoding function*. Furthermore, we will write $\langle x_1, \ldots, x_n \rangle$ in place of $\langle x_1 \rangle^\frown \langle x_2 \rangle^\frown \ldots^\frown \langle x_n \rangle$, and we will say that $\langle \ldots \rangle$ is a *coding function*. Note that $|\langle x_1, \ldots, x_n \rangle| = n$ and $(\langle x_1, \ldots, x_n \rangle)_i = x_i$ (for $i = 1, \ldots, n$).

It is important that our coding system enjoys the monotonicity properties given by the next lemma.

Lemma 7.3.16. *For any sequence $a_1, \ldots, a_n, a_{n+1}$ of natural numbers, we have*

$$\langle a_1, \ldots, a_i, \ldots, a_n \rangle \ < \ \langle a_1, \ldots, a_i + 1, \ldots a_n \rangle$$

and

$$\langle a_1, \ldots, a_n \rangle \ < \ \langle a_1, \ldots, a_n, a_{n+1} \rangle \ .$$

These monotonicity properties guarantee that

- all the numbers in the sequence encoded by the number x will be smaller than x
- the code for a subsequence of a sequence will be smaller than the code for the sequence itself.

This makes it easy to find primitive recursive definitions of predicates and functions dealing with encoded sequences. For example, the predicate

$$\text{code}(x) \ \wedge \ (\forall i \leq |x|) \, [\, 0 < i \rightarrow (\exists a \leq x)(\exists b \leq x) \, [\, (x)_i = \langle a, b \rangle \,] \,]$$

states that x encodes a sequence of pairs. The predicate

$$\text{code}(x) \ \wedge \ \text{code}(y) \wedge \\ (\exists v \leq x) \, (\exists w \leq x) \, [\, \text{code}(v) \ \wedge \ \text{code}(w) \ \wedge \ v^\frown y^\frown w = x \,]$$

states that y encodes a subsequence of the sequence encoded by x. If

$$f(x, i) \ = \ (\mu y \leq x) \, \big[\, (\exists z \leq x) \, [\, \text{code}(y) \ \wedge \ \text{code}(z) \ \wedge \ y^\frown z = x \ \wedge \ |y| = i \,] \, \big]$$

then we have $f(\langle x_1, \ldots x_n \rangle, i) = \langle x_1, \ldots x_i \rangle$ when $i \leq n$.

We round off this section on primitive recursion by a lemma on Gödel numbering of $\mathcal{L}_{NT}$-formulas. We will need this lemma to prove some of our main results.

Lemma 7.3.17. *Let $\phi(x)$ be a $\mathcal{L}_{NT}$-formula. There exists a primitive recursive function f_ϕ such that $f_\phi(a) = \ulcorner \phi(\overline{a}) \urcorner$*

Proof. Let $g(0) = \langle 9 \rangle$ and $g(y+1) = \langle 11, g(y) \rangle$. Then, we have $g(a) = \ulcorner \overline{a} \urcorner$ for every natural number a by Definition 5.7.1. The function g is primitive recursive by the lemmas above.

We define the function f_t by induction on the structure of the $\mathcal{L}_{NT}$-term t. Let

- $f_t(a) = g(a)$ if t is the variable x
- $f_t(a) = \langle 2 \cdot i \rangle$ if t is the variable v_i and v_i is different from x
- $f_t(a) = \langle 9 \rangle$ if t is 0
- $f_t(a) = \langle 11, f_{t_1}(a) \rangle$ if t is St_1
- $f_t(a) = \langle 13, f_{t_1}(a), f_{t_2}(a) \rangle$ if t is $+t_1t_2$
- $f_t(a) = \langle 15, f_{t_1}(a), f_{t_2}(a) \rangle$ if t is $\cdot t_1t_2$
- $f_t(a) = \langle 17, f_{t_1}(a), f_{t_2}(a) \rangle$ if t is Et_1t_2.

The function f_t is primitive recursive by the lemmas above. By Definition 5.7.1, we have $f_t(a) = \ulcorner t\frac{x}{\overline{a}} \urcorner$ where $t\frac{x}{\overline{a}}$ denotes the term t where each occurrence of the variable x is replaced by the numeral $\overline{a}$.

Now it is straightforward to define a primitive recursive function f_ϕ that satisfies the lemma. Define the function by induction on the structure of the formula ϕ (see Exercise 6). □

7.3.1 Exercises

1. (a) The signum function sg is given by

$$\mathrm{sg}(x) = \begin{cases} 0 & \text{if } x = 0 \\ 1 & \text{otherwise.} \end{cases}$$

   Prove that the signum function is primitive recursive.

   (b) Let $\sharp(x, y, z) = y$ if $x = 0$, and let $\sharp(x, y, z) = z$ if $x \neq 0$. Let $\max(x, y) = x$ if $x \geq y$, and let $\max(x, y) = y$ if $x < y$. Prove that $\sharp$ and max are primitive recursive functions.

2. Show that the class of primitive recursive functions is closed under *definition by cases*: Let g_1, g_2 be primitive recursive functions, let P be a primitive recursive predicate, and let

$$f(x_1, \ldots, x_n) = \begin{cases} g_1(x_1, \ldots, x_n) & \text{if } P(x_1, \ldots, x_n) \\ g_2(x_1, \ldots, x_n) & \text{otherwise.} \end{cases}$$

   Prove that f is a primitive recursive function.

3. (a) The predicate $D(x, y)$ states that x is a proper divisor of y. Prove that this predicate is primitive recursive (e.g., the proper divisors of 14 are 1,2, and 7).

   (b) A *perfect number* is a natural number that is equal to the sum of its proper divisors. For example, 6 is perfect because $6 = 1+2+3$, and 28 is perfect because $28 = 1 + 2 + 4 + 7 + 14$. The two next perfect numbers are 496 and 8128. Prove that the set of all perfect numbers is a primitive recursive set (it is not known if this set is infinite). Hint: Use the functions $p(i)$ and $\pi_i(m)$ to code and decode a sequence of numbers (see Lemma 7.3.12 and 7.3.13).

4. The numbers in the sequence $1, 1, 2, 3, 5, 8, 13, \ldots$ are known as the Fibonacci numbers. The n^{th} number in the sequence a_n is given by $a_0 = a_1 = 1$ and $a_{n+2} = a_n + a_{n+1}$. Prove that the function that yields the n^{th} Fibonacci number is primitive recursive.

   Careful! The Fibonacci numbers are defined by a recursion of the form

   $$f(n+2) \quad = \quad \ldots f(n+1) \ldots f(n) \ldots$$

   while the recursion in our primitive recursive definition scheme is of the form

   $$f(n+1) \quad = \quad \ldots f(n) \ldots \ .$$

   Hint: Use the functions $p(i)$ and $\pi_i(m)$ to code and decode a sequence of Fibonacci numbers (see Lemma 7.3.12 and 7.3.13).

5. (a) Give a full primitive recursive definition of the multiplication function: Set up equations that define the function g such that $g(x, y) = x \cdot y$. You need to define several other functions before you can define the function g. Set up equations that define these functions. Each function shall either be defined with an equation of the form

   $$f(x_1, \ldots, x_n) = h(g_1(x_1, \ldots, x_n), \ldots, g_m(x_1, \ldots, x_n)) \qquad \text{(Composition)}$$

   or with two equations of the form

   $$\begin{aligned} f(x_1, \ldots, x_n, 0) &= g(x_1, \ldots, x_n) \\ f(x_1, \ldots, x_n, y+1) &= h(x_1, \ldots, x_n, y, f(x_1, \ldots, x_n, y)) \ . \end{aligned} \qquad \text{(Primitive Recursion)}$$

   (b) The factorial function $n!$ is given by $0! = 1$ and $(n+1)! = (n+1) \cdot n!$. Give a full primitive recursive definition of the factorial function.

6. Complete the proof of Lemma 7.3.17. Define the function f_ϕ by induction on the structure of the $\mathcal{L}_{NT}$-formula ϕ.

7. Prove that every set definable by a Δ-formula is primitive recursive, that is, prove for any Δ-formula $\phi(\underset{\sim}{x})$, there exists a primitive recursive relation $R(\underset{\sim}{x})$ such that $R(\underset{\sim}{a})$ holds if, and only, if $\mathfrak{N} \models \phi(\underset{\sim}{\overline{a}})$.

   There exist primitive recursive sets that are not Δ-definable. So the class of Δ-definable set does not coincide with the class of primitive recursive relations. However, it is hard to find a natural example of a primitive recursive set that is not Δ-definable. Such sets do rarely occur in standard mathematics (extra bonus points for finding an example on your own).

8. We define what it means that a primitive recursive function f *is of rank* n:

   - The initial functions ($\mathcal{S}$, $\mathcal{I}_i^n$, $\mathcal{O}$) are of rank 0.
   - f is of rank n if

   $$f(\underset{\sim}{x}) = h(g_1(\underset{\sim}{x}), \ldots, g_m(\underset{\sim}{x})) \qquad \text{(Composition)}$$

   where $g_1, \ldots, g_m$ and h are of ranks less or equal to n.

   - f is of rank $n+1$ if

   $$\begin{array}{rcl} f(\underset{\sim}{x}, 0) & = & g(\underset{\sim}{x}) \\ f(\underset{\sim}{x}, y+1) & = & h(\underset{\sim}{x}, y, f(\underset{\sim}{x}, y)) \end{array} \qquad \text{(Primitive Recursion)}$$

   where g and h are of ranks less or equal to n.

   Note that any function of rank n also will be of rank $n+1$, but some functions of rank $n+1$ will not be of rank n.

   (a) Let f be a function of rank 0. Prove that there exist numbers $i, k \in \mathbb{N}$ such that $f(x_1, \ldots, x_n) \leq x_i + k$.

   (b) Let f be a function of rank 1. Prove that there exists some $k \in \mathbb{N}$ such that

   $$f(x_1, \ldots, x_n) \ \leq \ k(\max(x_1, \ldots x_n) + 1) \ .$$

   (c) Give examples of some primitive recursive that are not of rank 0 or 1. Prove that the multiplication function is not of rank 1. Is 2^x a function of rank 2?

9. We define the *Ackermann function* A by

   - $A(0, x) = x + 2$
   - $A(y+1, 0) = 2$
   - $A(y+1, x+1) = A(y, A(y+1, x))$.

(You will find several versions of the Ackermann function in the literature.) The type of recursion used to define the Ackermann function is called *double recursion*. Double recursion cannot be reduced to primitive recursion.

(a) Prove that the Ackermann function is total. Prove that it is monotone in both arguments. So prove that $A(y,x) \leq A(y+1,x)$, and prove that $A(y,x) \leq A(y,x+1)$. Furthermore, prove that $A(1,x) = 2x+2$, and prove that $2^x < A(2,x)$. How fast will the function $A(3,x)$ grow? How fast will $A(4,x)$ grow?
For each $n \in \mathbb{N}$, we define the function A_n by $A_n(x) = A(n,x)$. The function A_n is called the *n-branch of the Ackermann function*.

(b) Prove that A_n is primitive recursive (for any $n \in \mathbb{N}$).
Let A_n^k denote the k^{th} iterate of A_n, that is, $A_n^0(x) = x$ and $A_n^{k+1}(x) = A_n(A_n^k(x))$.

(c) Let f be a function of rank n (see Exercise 8). Prove that there exists some $k \in \mathbb{N}$ such that

$$f(x_1,\ldots,x_\ell) \;\leq\; A_n^k(\max(x_1,\ldots,x_\ell)) \,.$$

(d) Prove that the Ackermann function is not primitive recursive. (Hint: Let $d(x) = A_x^x(x)$. Then, d is primitive recursive if A is. For any $k,n \in \mathbb{N}$, we have $A_n^k(x) < d(x)$ for all sufficiently large x.)

7.4 Computable Functions and Computable Indices

The time has come to introduce the computable indices. A computable index for a function f is a natural number. By decoding this number we can see how the function can be defined by applying our formal definition schemes, and thus, we can extract an algorithm for computing the function. From this point of view, a computable index is a program that can be executed by a computer.

It is also fair to say that computable indices are Gödel numbers. This may be confusing of course as some (for example, anyone who has read through Section 5.7) will expect Gödel numbers to encode first-order formulas, derivations in a formal calculus, and that kind of stuff. But a computable index is nothing but a Gödel number that encodes the definition of a computable function. Note that each clause of the definition below corresponds to a clause of Definition 7.2.1 and that the function $\langle\cdot\rangle$ is the same coding function as in Section 7.3 and Chapter 4.

Definition 7.4.1. The number e is a *computable index for the function f* if one of the following clauses holds:

(1) $e = \langle 0 \rangle$ and f is the successor function $\mathcal{S}$

(2) $e = \langle 1, n, i \rangle$ and f is the projection function $\mathcal{I}_i^n$

(3) $e = \langle 2 \rangle$ and f is the zero function $\mathcal{O}$

(4) $e = \langle 3, n, b_1, \ldots b_m, a \rangle$ and

$$f(x_1, \ldots, x_n) \;=\; h(g_1(x_1, \ldots, x_n), \ldots, g_m(x_1, \ldots, x_n))$$

and $b_1, \ldots b_m, a$ are indices for, respectively, $g_1, \ldots, g_m, h$.

(5) $e = \langle 4, n, a, b \rangle$ and

- $f(x_1, \ldots, x_n, 0) = g(x_1, \ldots, x_n)$
- $f(x_1, \ldots, x_n, y+1) = h(x_1, \ldots, x_n, y, f(x_1, \ldots, x_n, y))$

and a, b are indices for, respectively, g, h.

(6) $e = \langle 5, n, a \rangle$ and

$$f(x_1, \ldots, x_n) \;=\; (\mu i)[g(x_1, \ldots, x_n, i)]$$

and a is an index for g.

Occasionally, we will say that *the function f has index e* when e is a computable index for f.

It is important to understand the relationship between the computable functions and the computable indices. A function has infinitely many indices, whereas an index will be assigned to one, and only one, function. The reader might find it enlightening to study the proof of the next lemma. The lemma is a weak version of the well known Padding Lemma (see Exercise 4 for the full version).

Lemma 7.4.2 (Padding Lemma). *Every computable function has infinitely many indices.*

Proof. Let f be a computable function. We will first prove that

$$\text{there exists } e \in \mathbb{N} \text{ such that } e \text{ is an index for } f \qquad (*)$$

by induction on the structure of f. Our proof will have one case for each of the six clauses of Definition 7.2.1. In each of these cases, we use the corresponding clause of Definition 7.4.1 to conclude that f has an index.

Case (1): f is the function $\mathcal{S}$. Then, by Clause (1) of Definition 7.4.1, the number $\langle 0 \rangle$ is an index for f.

Case (2): f is the function $\mathcal{I}_i^n$. Then, by Clause (2) of Definition 7.4.1, the number $\langle 1, n, i \rangle$ is an index for f.

Case (3): f is the function $\mathcal{O}$. Then, by Clause (3) of Definition 7.4.1, the number $\langle 2 \rangle$ is an index for f.

Case (4): $f(x_1, \ldots, x_n) = h(g_1(x_1, \ldots, x_n), \ldots, g_m(x_1, \ldots, x_n))$. By the induction hypothesis, we have indices $b_1, \ldots b_m, a$ for, respectively, the functions $g_1, \ldots, g_m, h$. By Clause (4) of Definition 7.4.1, the number $\langle 3, n, b_1, \ldots b_m, a \rangle$ is an index for f.

The case when $f(\underset{\sim}{x}, 0) = g(\underset{\sim}{x})$ and $f(\underset{\sim}{x}, y+1) = h(\underset{\sim}{x}, y, f(\underset{\sim}{x}, y))$ is similar to (4). So is the case when $f(\underset{\sim}{x}) = (\mu i)[g(\underset{\sim}{x}, i)]$. This proves (*).

Next we prove that each computable function has not just one, but infinitely many indices. Let f be a computable function, and let e be an index for f. Such an e exists by (*). We define a sequence of functions $f_0, f_1, f_2, \ldots$ and a sequence of indices $e_0 < e_1 < e_2 < \ldots$ such that

- e_i is an index for f_i
- f_i is the same function as f_{i+1}.

Let f_0 be the function f, and let e_0 be the index e. Let $f_{i+1}(x_1, \ldots x_n) = \mathcal{I}^1_1(f_i(x_1, \ldots x_n))$. Then, f_{i+1} is a computable function by Clause (4) of Definition 7.2.1. Moreover, $f_{i+1}(\underset{\sim}{x}) = f_i(\underset{\sim}{x})$ since $\mathcal{I}^1_1(y) = y$. Let $e_{i+1} = \langle 3, n, e_i, \langle 1, 1, 1 \rangle \rangle$. Then, e_{i+1} is an index for f_{i+1} by Clause (2) and Clause (4) of Definition 7.4.1. Hence, every number in sequence $e_0, e_1, e_2, \ldots$ is an index for f. Obviously we have $e_i < e_{i+1}$, and thus we conclude that f has infinitely many indices. □

The next theorem introduces the predicate $\mathcal{T}_n$ (for any $n \geq 1$) and a function $\mathcal{U}$. The predicate is known as the *Kleene T-predicate*. The exact definitions of $\mathcal{T}_n$ and $\mathcal{U}$ will be given when we prove the theorem. All the properties of $\mathcal{T}_n$ and $\mathcal{U}$ that we need outside the proof are given by the theorem.

Theorem 7.4.3 (Kleene's Normal Form Theorem). *Let $n \in \mathbb{N}$. There exist a primitive recursive predicate $\mathcal{T}_n$ and a primitive recursive function $\mathcal{U}$ such that*

- *if e is not an index for an n-ary function, then $\mathcal{T}_n(e, x_1, \ldots x_n, t)$ does not hold (for all $x_1, \ldots x_n, t \in \mathbb{N}$)*
- *if e is an index for the n-ary function f, then*

$$f(x_1, \ldots x_n) \text{ is defined} \quad \Leftrightarrow \quad \text{there exists } t \text{ such that } \mathcal{T}_n(e, x_1, \ldots x_n, t) \text{ holds}$$

- *whenever $f(x_1, \ldots x_n)$ is defined and e is an index for f*

$$f(x_1, \ldots x_n) = \mathcal{U}((\mu t)[\mathcal{T}_n(e, x_1, \ldots x_n, t)]) .$$

Kleene's Normal Form Theorem is one of the cornerstones of computability theory. The idea behind the proof of this theorem should be easy to grasp for those who are familiar with coding and Gödel numbering: The Kleene T-predicate $\mathcal{T}_n(e, x_1, \ldots x_n, t)$ states that the number t encodes an execution of the program given by the index e on the inputs $x_1, \ldots x_n$. The function $\mathcal{U}(t)$ picks the output from the computation encoded by t. However, the details of the proof are complicated and involved, and we postpone the proof until the next section. That section may be skipped on a first reading.

We should be careful when we write equality signs between expressions involving computable functions as some functions will be undefined for some arguments. Some authors use the symbol $\simeq$ and write $A \simeq B$ when they want to state that either

$$\text{both } A \text{ and } B \text{ are defined and equal the same natural number} \tag{1}$$

or

$$\text{both } A \text{ and } B \text{ are undefined.} \tag{2}$$

We will use the standard equality sign and write $A = B$ when we want to state that either (1) or (2) hold. This is not unproblematic. Our notation is ambiguous, e.g., we also use $=$ to denote the primitive recursive equality relation (see Lemma 7.3.5). So the reader should think twice when an equality sign appears, but we do not think our choice of notation will cause too much trouble.

Definition 7.4.4. For each $e \in \mathbb{N}$, we define the *e-th computable function* of arity n, written $\{e\}^n$, by

$$\{e\}^n(x_1, \ldots, x_n) \;=\; \mathcal{U}((\mu t)[\mathcal{T}_n(e, x_1, \ldots x_n, t)])$$

where the function $\mathcal{U}$ and predicate $\mathcal{T}_n$ are given by Kleene's Normal Form Theorem (Theorem 7.4.3).

> *Chaff:* It might be worth noting that the e's in the above definition come in two flavors. Sometimes e is an index of an n-ary computable function, as defined in Definition 7.4.1. If e is such a number, then $\{e\}^n$ is exactly the function that you would expect: the function that has index e. But sometimes (actually, most of the time) e is not, strictly speaking, an index of an n-ary computable function. In this case $\{e\}^n$ is the function that is defined nowhere. So even though e is not the index of a computable function, $\{e\}^n$ is a computable function. And you thought this stuff was easy...

Theorem 7.4.5 (Enumeration Theorem). *The sequence*

$$\{0\}^n, \{1\}^n, \{2\}^n, \ldots$$

is an enumeration of the computable functions of arity n in the sense that

(1) for each $e \in \mathbb{N}$, the function $\{e\}^n$ is computable

(2) for each computable function f of arity n, there exists e such that

$$f(x_1, \ldots, x_n) \quad = \quad \{e\}^n(x_1, \ldots, x_n) \, .$$

Moreover, this enumeration is computable in the sense that

(3) there exists a computable function g of arity $n+1$ such that

$$g(e, x_1, \ldots, x_n) \quad = \quad \{e\}^n(x_1, \ldots, x_n) \, .$$

Proof. Definition 7.4.4 states that

$$\{e\}^n(x_1, \ldots, x_n) = \mathcal{U}((\mu t)[\mathcal{T}_n(e, x_1, \ldots x_n, t)]) \, .$$

Now, $\mathcal{U}$ is a computable function, and $\mathcal{T}_n$ is a computable predicate. Moreover, the class of computable functions is closed under composition and minimalization. Thus, $\{e\}^n$ is a computable function. This proves (1).

Let f be a computable function. By Lemma 7.4.2, there exists a number $\hat{e}$ which is a computable index for the function f. We have

$$f(x_1, \ldots x_n) \quad = \quad \mathcal{U}((\mu t)[\mathcal{T}_n(\hat{e}, x_1, \ldots x_n, t)]) \quad = \quad \{\hat{e}\}^n(x_1, \ldots, x_n) \, .$$

The first equality holds by Kleene's Normal Form Theorem, and the second equality holds by Definition 7.4.4. This completes the proof of (2).

We turn to the proof of (3). Let

$$g(y, x_1, \ldots, x_n) = \mathcal{U}((\mu t)[\mathcal{T}_n(y, x_1, \ldots x_n, t)]) \, .$$

Now, $\mathcal{U}$ is a computable function and $\mathcal{T}_n$ is a computable predicate. Moreover, the class of computable functions is closed under composition and minimalization. Thus, g is a computable function. By Definition 7.4.4, we have $g(e, x_1, \ldots, x_n) = \{e\}^n(x_1, \ldots, x_n)$. □

A couple of important results are easy corollaries of the Enumeration Theorem. The universal function u in our first corollary is a mathematical model of the programmable computer. You feed the function with two inputs. One of the inputs is a program e that you want the computer to execute. The other input is the input that you want e to be executed on. The successful development of computability theory in the 1930s preceded and inspired the engineering of computing machinery. The first programmable real-life computers came into being during the Second World War.

Corollary 7.4.6 (Universal Function Theorem). *There exists a computable function $u : \mathbb{N}^2 \to \mathbb{N}$ with the following property: for any n-ary computable function f there exists $e \in \mathbb{N}$ such that*

$$u(e, \langle x_1, \ldots, x_n \rangle) = f(x_1, \ldots, x_n)$$

for all $x_1, \ldots, x_n \in \mathbb{N}$. Moreover, $u(y, x) = \{y\}^1(x)$.

Proof. Let $u(y, x) = \{y\}^1(x)$. Let f be any computable function, and let e be an index for the function f_0 given by $f_0(z) = f((z)_1, \ldots, (z)_n)$. Then,

$$\begin{aligned} u(e, \langle x_1, \ldots, x_n \rangle) &= \{e\}^1(\langle x_1, \ldots, x_n \rangle) = f_0(\langle x_1, \ldots, x_n \rangle) = \\ & f((\langle x_1, \ldots, x_n \rangle)_1, \ldots, (\langle x_1, \ldots, x_n \rangle)_n) = f(x_1, \ldots, x_n) \,. \end{aligned}$$

□

There is another straightforward consequence of the Enumeration Theorem: we can diagonalize and find a total function that is not computable. From the enumeration $\{0\}^1, \{1\}^1, \{2\}^1, \ldots$ we define a diagonal function d such that $d(0) \neq \{0\}^1(0)$ and $d(1) \neq \{1\}^1(1)$ and $d(2) \neq \{2\}^1(2)$, and so on. In general we define d such that $d(i) \neq \{i\}^1(i)$ for every $i \in \mathbb{N}$. Then d cannot be computable as d cannot appear in the enumeration:

Corollary 7.4.7. *The function d given by*

$$d(x) = \begin{cases} \{x\}^1(x) + 1 & \text{if } \{x\}^1(x) \text{ is defined} \\ 0 & \text{if } \{x\}^1(x) \text{ is undefined.} \end{cases}$$

is not computable.

Proof. Assume that d is computable. By the Enumeration Theorem we have $e \in \mathbb{N}$ such that

$$d(x) = \{e\}^1(x) \text{ for all } x. \qquad (*)$$

Now, what happens when we apply the function d to this e? We will consider two cases: the case when $\{e\}^1(e)$ is defined and the case when $\{e\}^1(e)$ is undefined. In both cases we will encounter a contradiction, and thus d cannot be computable.

Assume $\{e\}^1(e)$ is defined. We have $d(e) = \{e\}^1(e)$ by (*), and we have $d(e) = \{e\}^1(e) + 1$ by the definition of d. But then $\{e\}^1(e) = \{e\}^1(e) + 1$, and we have arrived at a contradiction.

Assume $\{e\}^1(e)$ is undefined. By (*), $d(e)$ is undefined. By the definition of d, we have $d(e) = 0$. It is a contradiction that $d(e)$ equals 0 when $d(e)$ is undefined. □

Maybe the last corollary does not seem very interesting at a first sight. Diagonalization is a familiar proof technique, and it should be no surprise that a non-computable diagonal function like d exists once we have an enumeration of all the computable functions.

> *Chaff:* Well, maybe it is fair to say that diagonalization is familiar to you now, after you have seen it a couple of times. Probably the random person you meet on the street is not familiar with it...

But the corollary does take us one step closer to a proof of the next theorem. The theorem states that the famous *Halting Problem* is undecidable, that is, no algorithm can decide if a computer executing a program e on some input x ever will be finished. That is definitely interesting and non-trivial! The idea behind the proof, however, is not difficult. Quite simply we will show that if the Halting Problem is decidable, then the diagonal function d in Corollary 7.4.7 is computable. But d is not computable. Hence, the Halting Problem is not decidable.

Theorem 7.4.8 (Undecidability of the Halting Problem). *Let u be the universal function given in Corollary 7.4.6, that is, let $u(y,x) = \{y\}^1(x)$. Let H be the predicate given by*

$$H(y,x) \quad \Leftrightarrow \quad u(y,x) \text{ is defined}$$

Then, H is not a computable predicate.

Proof. Assume for the sake of a contradiction that H is computable, and so χ_H, the characteristic function of H, is computable. Recall that $u(y,x) = \{y\}^1(x)$. Thus, $\{y\}^1(x)$ is defined when $\chi_H(y,x) = 0$, and $\{y\}^1(x)$ is undefined when $\chi_H(y,x) = 1$. Let

$$d(x) \;=\; \begin{cases} \{x\}^1(x) + 1 & \text{if } \chi_H(x,x) = 0 \text{ (i.e. if } \{x\}^1(x) \text{ is defined)} \\ 0 & \text{if } \chi_H(x,x) = 1 \text{ (i.e. if } \{x\}^1(x) \text{ is undefined).} \end{cases}$$

Then, d is computable as χ_H is computable. But this is a contradiction as d is nothing but the diagonal function from Corollary 7.4.7 – the function we have proven not to be computable. □

Both the Halting Problem and the Entscheidungsproblem were proved to be undecidable by Alan M. Turing in his 1937 paper *On Computable Numbers, with an Application to the Entscheidungsproblem* [Turing 37]. The Turing machine was introduced in the same paper. Turing used the undecidability of the Halting Problem in his proof of the undecidability of the Entscheidungsproblem.

This section's last theorem does perhaps not seem very interesting per se, and unfortunately its proof is not a thing of beauty. However, it is an important technical result that is needed to prove many deep theorems of computability theory. We will need the theorem later (e.g., in our proof a strong version of Gödel's First Incompleteness Theorem and in our proof of Rice's Theorem).

Theorem 7.4.9 (The S-m-n Theorem). *Let $m, n \in \mathbb{N}$. There exists a primitive recursive function S_n^m (of arity $n+1$) such that*

$$\{S_n^m(e, x_1, \ldots, x_n)\}^m(y_1, \ldots y_m) \;=\; \{e\}^{n+m}(x_1, \ldots x_n, y_1, \ldots y_m)\,.$$

Proof. Let h be a computable function of arity $n+m$, let $a_1, \ldots, a_n$ be fixed natural numbers, and let $c_{a_1}^m, \ldots, c_{a_n}^m$ be the primitive recursive constant functions given in Lemma 7.3.1. From the function h, the functions $c_{a_1}^m, \ldots, c_{a_n}^m$ and the projection functions $\mathcal{I}_1^m, \mathcal{I}_2^m, \ldots, \mathcal{I}_m^m$, we define a function f by a single application of the composition scheme. Let

$$f(\underset{\sim}{y}) \;=\; h(c_{a_1}^m(\underset{\sim}{y}), \ldots, c_{a_n}^m(\underset{\sim}{y}), \mathcal{I}_1^m(\underset{\sim}{y}), \mathcal{I}_2^m(\underset{\sim}{y}), \ldots, \mathcal{I}_m^m(\underset{\sim}{y})) \qquad (*)$$

where $\underset{\sim}{y} = y_1, \ldots, y_m$. Obviously, we have

$$\begin{aligned} f(y_1, \ldots, y_m) &= h(a_1, \ldots, a_n, y_1, \ldots, y_m) \\ &= \{e\}(a_1, \ldots, a_n, y_1, \ldots, y_m) \qquad (**) \end{aligned}$$

when e is an index for h.

We need the following claim:

> **(Claim)** For each $m \in \mathbb{N}$, there exists a primitive recursive function q_m such that $q_m(x)$ yields a computable index for the function c_x^m.

The proof of this claim is given as an exercise (Exercise 6).

Let q_m be the primitive recursive function given by the claim. Thus, $q_m(a_i)$ is an index for $c_{a_i}^m$ (for $i = 1, \ldots, n$). Clause (2) of Definition 7.4.1 says that $\langle 1, m, j\rangle$ is an index for the function $\mathcal{I}_j^m$ (for $j = 1, \ldots m$). Let e be an index for h. By (*) and Clause (4) of Definition 7.4.1, the number

$$\langle 3, m, q_m(a_1), \ldots, q_m(a_n), \langle 1, m, 1\rangle, \ldots, \langle 1, m, m\rangle, e\rangle$$

is an index for f. Thus let

$$S_n^m(e, x_1, \ldots x_n) = \langle 3, m, q_m(x_1), \ldots, q_m(x_n), \langle 1, m, 1\rangle, \ldots, \langle 1, m, m\rangle, e\rangle\,.$$

The function S_n^m is primitive recursive by the lemmas in Section 7.3. Furthermore

$$\begin{aligned} \{S_n^m(e, x_1, \ldots x_n)\}(y_1, \ldots, y_m) \;=\; f(y_1, \ldots, y_m) \;=\; \\ \{e\}(x_1, \ldots, x_n, y_1, \ldots, y_m)\,. \end{aligned}$$

The first equality holds since $S_n^m(e, x_1, \ldots x_n)$ is an index for f, and the second equality holds by (**). □

7.4.1 Exercises

1. We define the function h by

$$h(x, y, z) = \mathcal{S}(\mathcal{I}_3^3(x, y, z))$$

and we define the function f by

$$\begin{aligned} f(x, 0) &= \mathcal{I}_1^1(x) \\ f(x, y+1) &= h(x, y, f(x, y)) \end{aligned}$$

Thus, h is defined by the scheme of composition, and f is defined by the scheme of primitive recursion. Give a computable index for h. Give a computable index for f.

2. Give three computable indices for the function f where $f(x) = x + 2$.

3. The definition scheme

$$f(\underset{\sim}{x}) = \begin{cases} g_1(\underset{\sim}{x}) & \text{if } h(\underset{\sim}{x}) = 0 \\ g_2(\underset{\sim}{x}) & \text{if } h(\underset{\sim}{x}) = n \text{ and } n \in \mathbb{N} \text{ and } n \neq 0 \\ \text{undefined} & \text{if } h(\underset{\sim}{x}) \text{ is undefined} \end{cases}$$

shows how the function f is defined from the functions g_1, g_2, and h. This scheme is called *definition by cases*.

Prove that the computable functions are closed under definition by cases.

Be careful! Let $\sharp(x, y, z) = y$ if $x = 0$, and let $\sharp(x, y, z) = z$ if $x \neq 0$. Then, $\sharp$ is a primitive recursive function. If we define f by the composition $f(\underset{\sim}{x}) = \sharp(h(\underset{\sim}{x}), g_1(\underset{\sim}{x}), g_2(\underset{\sim}{x}))$, then $f(\underset{\sim}{x})$ will be undefined if, e.g., $g_2(\underset{\sim}{x})$ is undefined. Use Kleene's Normal Form Theorem to prove that the computable functions are closed under definition by cases.

4. This is the full version of the Padding Lemma:

> Every computable function has an index. Moreover, there exists a primitive recursive function *pad* with the following property: If e is a computable index for a function f, then $\{y \mid pad(e, i) = y \text{ and } i \in \mathbb{N}\}$ is an infinite set of indices for f.

Prove the full version.

5. The Ackermann function A and the n^{th} branch of the Ackermann function A_n are defined in Exercise 9 of Section 7.3.1. You proved, as part of that exercise, that A is not a primitive recursive function, even though A appears to be computable. Now you have the tools to prove that A is computable. Proceed the following way:

(a) argue that there exists a primitive recursive function f such that $f(n)$ yields a computable index for A_n, then

(b) use f and the fact that $A_y(x) = A(y, x)$ to prove that A is computable function.

6. Let c_i^n be the n-ary constant function from Lemma 7.3.1, that is,

$$c_i^n(x_1, \ldots, x_n) = i,$$

where i and n are fixed natural numbers. Thus, we have, e.g., $c_{17}^3(9, 2, 1) = 17$ and $c_{17}^2(0, 0) = 17$. Prove that (for each $n \in \mathbb{N}$) there exists a primitive recursive function q_n such that $q_n(x)$ yields a computable index for the function c_x^n.

7. Discuss the S-m-n Theorem. Give an informal explanation of the theorem. If we compare the index e to program written in real-life programming language like, e.g., Java or Caml, what should we compare $S_n^m(e, x_1, \ldots, x_n)$ to?

8. Let f be a binary computable function. Prove that there exists a primitive recursive function $\imath$ such that $\{\imath(y)\}(x) = f(y, x)$.

9. The following assertion is known as the Second Recursion Theorem:

> For any binary computable function f, there exists an index e such that
>
> $$\{e\}^1(x) \;=\; f(e, x)\,.$$

The reader should be aware that there are several versions of the Second Recursion Theorem and, of course, there is also a First Recursion Theorem.

(a) Discuss the Second Recursion Theorem. Give an informal explanation of the theorem.

(b) Prove the Second Recursion Theorem (this is a very hard exercise).

10. Can a computer program print its own code? It sounds like a contradiction that there should be such a program, but strangely enough such programs exist. It is indeed possible to write a program (e.g., in Java) that outputs it own (Java) code. An amusing allegory is given in Hofstadter [Hofstadter 85]:

> Alphabetize and append, copied in quote, these words: 'these append, in Alphabetize and words: quote, copied'

Carry out the instruction above and study the result of your work.

A computable index can be viewed as program code. Our mathematical model of a programmable computer is the universal function u given in Corollary 7.4.6 (recall that $u(y,x) = \{y\}^1(x)$). Prove that there exists a computable index e such that we have $u(e,x) = e$ for any x. It is surprisingly easy to prove this if you use the Second Recursion Theorem (see Exercise 9).

7.5 The Proof of Kleene's Normal Form Theorem.

We have been pretending to show you a perfect ball, but now are turning our ball around, and you will see a rusty spike sticking out. This spike is the proof of Kleene's Normal Form Theorem. When we turn the ball again, so that you do not see the spike anymore, you will soon forget that it is there.

Aesthetics is an essential part of mathematics. Beauty and perfection can be gained by pressing the rusty spikes into the ball. Sometimes we only partly succeed in hiding all the ugly spikes inside the ball: we end up with a ball where a few of them are sticking out. This is not so bad if these spikes are not visible when we watch the ball from a particular angle. If we just remember that we have a ball that is meant to be watched from a particular angle, we do not need to see anything but a perfect ball.

Why do we compare the proof of Kleene's Normal Form Theorem to a rusty spike? Well, the idea behind the proof is pretty easy to catch, but it is not easy to write up a decent and readable proof of the theorem. If you do not structure your arguments carefully, the details will soon become intolerably involved. And even if you structure your proof well, you still cannot avoid showing the reader some rather unpleasant formulas. It is tempting to argue to a certain point, and then, wave your hands and ask the reader to work out the details on his own. We have tried to resist that temptation.

We have assigned indices to the computable functions. Indices are natural numbers. By decoding an index, we can read off how a function f is defined by our formal definition schemes. Thus, an index for f provides enough information to compute f. To prove Kleene's Normal Form Theorem, we need to formalize how such a computation should be carried out. Our formalization is based on the idea that we can compute by writing down triplets, and a computation will be defined as a sequence of triplets, or more precisely, as a sequence of natural numbers that encode triplets. The triplets will be of the form $\langle e, \langle a_1, \ldots, a_n \rangle, b \rangle$ where

- e is a computable index

- $(a_1, \ldots, a_n)$ is a finite sequence of natural numbers, and n is the arity of the function to which e is assigned
- b is a natural number.

When we add a triplet $\langle e, \langle a_1, \ldots, a_n \rangle, b \rangle$ to a computation, we have established that b is the result of applying a function indexed by e to the arguments $a_1, \ldots, a_n$.

Some notation: We will use the capital Greek letters Γ and Ω to denote finite sequences of natural numbers, and we concatenate two such sequences by simply putting a comma between them. If $\Gamma = (c_1, \ldots, c_k)$ and $a \in \mathbb{N}$, then Γ, a denotes the sequence $\Gamma = (c_1, \ldots, c_k, a)$. Furthermore, $\Gamma \Vdash c$ denotes that the natural number c occurs (somewhere) in the sequence Γ.

> *Chaff:* Careful here! $\Vdash$ isn't the same as $\vdash$ or $\models$ of Chapters 1 and 2. There is no end of the ways that we can arrange vertical and horizontal lines to make up new symbols! Our choice is supposed to be suggestive, however. In many ways we want you to think of a formal computation as we define it here as something that is analogous to a formal deduction.

Definition 7.5.1. The collection of *computations* is the smallest set of finite sequences of natural numbers such that

($R0$) The empty sequence is a computation,

($R1$) $\Gamma, \langle \langle 0 \rangle, a, b \rangle$ is a computation if Γ is a computation and $b = a + 1$,

($R2$) $\Gamma, \langle \langle 1, n, i \rangle, \langle a_1, \ldots a_n \rangle, b \rangle$ is a computation if Γ is a computation and $1 \leq i \leq n$ and $b = a_i$

($R3$) $\Gamma, \langle \langle 2 \rangle, \langle \rangle, 0 \rangle$ is a computation if Γ is a computation ($\langle \rangle$ is the code for the empty sequence),

($R4$) $\Gamma, \langle \langle 3, n, d_1, \ldots d_m, d_0 \rangle, \langle a_1, \ldots a_n \rangle, b \rangle$ is a computation if Γ is a computation and there exist $v_1, \ldots, v_m \in \mathbb{N}$ such that

$$\Gamma \Vdash \langle d_i, \langle a_1, \ldots, a_n \rangle, v_i \rangle \qquad (\text{for } i = 1, \ldots, m)$$

and

$$\Gamma \Vdash \langle d_0, \langle v_1, \ldots, v_m \rangle, b \rangle,$$

($R5$) $\Gamma, \langle \langle 4, n, d_0, d_1 \rangle, \langle a_1, \ldots a_n, c \rangle, b \rangle$ is a computation if Γ is a computation and there exist $v_0, \ldots, v_c \in \mathbb{N}$ such that

$$\Gamma \Vdash \langle d_0, \langle a_1, \ldots a_n \rangle, v_0 \rangle$$

and

$$\Gamma \Vdash \langle d_1, \langle a_1, \ldots a_n, i, v_{i-1} \rangle, v_i \rangle \qquad (\text{for } i = 1, \ldots, c)$$

and $b = v_c$,

(*R*6) $\Gamma, \langle\langle 5, n, d\rangle, \langle a_1, \ldots a_n\rangle, b\rangle$ is a computation if Γ is a computation and there exist $v_0, \ldots, v_b \in \mathbb{N}$ such that

$$\Gamma \Vdash \langle d, \langle a_1, \ldots a_n, i\rangle, v_i\rangle \qquad \text{(for } i = 0, \ldots, b\text{)}$$

and $v_i \neq 0$ when $i < b$, and $v_b = 0$.

> *Chaff:* Our apologies. A computation, as defined above, isn't a computation in any reasonable sense of the word. A computation does, however, provide a code for a real computation, and that is a good thing! See the example that follows.

Note that each of the clauses (R1), (R2), (R3), (R4), (R5) and (R6) in the definition above corresponds to a clause of Definition 7.4.1, which again corresponds to a clause of Definition 7.2.1.

Let us look at an example. Let $h(x_1, x_2) = \mathcal{S}(\mathcal{I}_1^2(x_1, x_2))$, and let $f(x_1, x_2) = \mathcal{S}(h(x_1, x_2))$. Then, h is defined by the scheme of composition (over the functions $\mathcal{I}_1^2$ and $\mathcal{S}$), and f is defined by the scheme of composition (over the functions h and $\mathcal{S}$). Now

- $\langle 1, 2, 1\rangle$ is an index for $\mathcal{I}_1^2$
- $\langle 0\rangle$ is an index for $\mathcal{S}$
- $\langle\ 3\ ,\ 2\ ,\ \langle 1, 2, 1\rangle\ ,\ \langle 0\rangle\ \rangle$ is an index for h
- $\langle\ 3\ ,\ 2\ ,\ \langle 3, 2, \langle 1, 2, 1\rangle, \langle 0\rangle\rangle\ ,\ \langle 0\rangle\ \rangle$ is an index for f.

It is easy to see that $f(5, 92)$ equals 7. Below we give a computation Γ that formally establishes this. In the table below, the elements of the sequence Γ are in the second column. The third column gives a reference to the clause of Definition 7.5.1 that justifies adding the number to the computation. The comments in the first, fourth, and fifth columns should be self-explanatory.

1.	$\langle\ \langle 0\rangle\ ,\ 5\ ,\ 6\ \rangle$	(R1)		$\mathcal{S}(5) = 6$
2.	$\langle\ \langle 1, 2, 1\rangle\ ,\ \langle 5, 92\rangle\ ,\ 5\ \rangle$	(R2)		$\mathcal{I}_1^2(5, 92) = 5$
3.	$\langle\ \langle 3, 2, \langle 1, 2, 1\rangle, \langle 0\rangle\rangle\ ,\ \langle 5, 92\rangle\ ,\ 6\ \rangle$	(R4)	1,2	$h(5, 92) = 6$, where $h(x_1, x_2) = \mathcal{S}(\mathcal{I}_1^2(x_1, x_2))$
4.	$\langle\ \langle 0\rangle\ ,\ 6\ ,\ 7\ \rangle$	(R1)		$\mathcal{S}(6) = 7$
5.	$\langle\langle 3, 2, \langle 3, 2, \langle 1, 2, 1\rangle, \langle 0\rangle\rangle, \langle 0\rangle\rangle, \langle 5, 92\rangle, 7\rangle$	(R4)	3,4	$f(5, 92) = 7$, where $f(x_1, x_2) = \mathcal{S}(h(x_1, x_2))$

Of course, when you walk into a store that sells computations and you buy Γ, you're not going to get all of the extra stuff that we have provided above. All you will get is the computation—the sequence of numbers

$$\Gamma = \big(\langle\langle 0\rangle, 5, 6\rangle, \langle\ \langle 1, 2, 1\rangle, \langle 5, 92\rangle, 5\ \rangle, \ldots, \langle\langle 3, 2, \langle 3, 2, \langle 1, 2, 1\rangle, \langle 0\rangle\rangle, \langle 0\rangle\rangle, \langle 5, 92\rangle, 7\rangle\big)\,.$$

If Γ is the computation above and $e = \langle 3, 2, \langle 3, 2, \langle 1, 2, 1\rangle, \langle 0\rangle\rangle, \langle 0\rangle\rangle$, (so e is an index for the function f) then we have $\Gamma \Vdash \langle e, \langle 5, 92\rangle, 7\rangle$. We also have $\Omega \Vdash \langle e, \langle 5, 92\rangle, 7\rangle$ for any computation Ω extending Γ. Note that

$$\Gamma \Vdash \langle e, \langle a_1, \ldots, a_n\rangle, b\rangle$$

can be read as "the computation Γ shows that b is the result of applying the function with index e to the arguments $a_1, \ldots a_n$".

Lemma 7.5.2. *Let e be a computable index for the function f. Then, $f(a_1, \ldots, a_n) = b$ if and only if there exists a computation Γ such that $\Gamma \Vdash \langle e, \langle a_1, \ldots, a_n\rangle, b\rangle$.*

Proof. We prove the lemma by induction on the structure of the index e. The possible forms of e are given by Definition 7.4.1. Each of the six clauses of the definition corresponds to a case in our proof.

We start with the case when $e = \langle 0\rangle$. In this case f is the successor function $\mathcal{S}$. Assume $f(a) = b$. Then, as f is the successor function, we have $b = a+1$. Let Ω be any computation. Then $\Omega, \langle\langle 0\rangle, \langle a\rangle, b\rangle$ is a computation by Clause (R1) of Definition 7.5.1. Moreover, we have $\Omega, \langle\langle 0\rangle, \langle a\rangle, b\rangle \Vdash \langle\langle 0\rangle, \langle a\rangle, b\rangle$. This proves the "only if" direction. To prove the converse implication, assume that $\Gamma \Vdash \langle\langle 0\rangle, \langle a\rangle, b\rangle$. Then, the number $\langle\langle 0\rangle, \langle a\rangle, b\rangle$ occurs in the computation Γ. By inspecting Definition 7.5.1, we can verify that we have $b = a + 1$ whenever a number of the form $\langle\langle 0\rangle, \langle a\rangle, b\rangle$ occurs in a computation. Thus, as f is the successor function, we have $f(a) = b$.

The case when the index e is of the form $\langle 0\rangle$ is a base case where we do not need the induction hypotheses. There are two more base cases. The case when $e = \langle 1, n, i\rangle$ (and f is the projection function $\mathcal{I}_i^n$), and the case when $e = \langle 2\rangle$ (and f is the zero function $\mathcal{O}$). The proofs for these cases are similar to the proof for the case when $e = \langle 0\rangle$, and we omit the details.

We turn to the case when $e = \langle 3, n, d_1, \ldots d_m, d_0\rangle$. Then,

$$f(x_1, \ldots, x_n) \;=\; h(g_1(x_1, \ldots, x_n), \ldots, g_m(x_1, \ldots, x_n))$$

and $d_1, \ldots d_m, d_0$ are indices for $g_1, \ldots, g_m, h$, respectively. First we will prove that $f(a_1, \ldots, a_n) = b$ entails that there exists a computation Γ such that $\Gamma \Vdash \langle e, \langle a_1, \ldots, a_n\rangle, b\rangle$. Thereafter we will prove the converse implication.

Assume $f(a_1, \ldots, a_n) = b$. Then, there exists $v_1, \ldots, v_m \in \mathbb{N}$ such that

$$g(a_1, \ldots, a_n) = v_\imath \qquad (\text{for } \imath = 1, \ldots, m)$$

and $h(v_1, \ldots, v_m) = b$. By our induction hypothesis, we have computations $\Omega^\imath$ (for $\imath = 1, \ldots, m$) and Ω^0 such that

$$\Omega^\imath \Vdash \langle d_\imath, \langle a_1, \ldots a_n\rangle, v_\imath\rangle \qquad (\text{for } \imath = 1, \ldots, m)$$

and

$$\Omega^0 \Vdash \langle d_0, \langle v_1, \ldots v_m\rangle, b\rangle \ .$$

Let Γ be the sequence that is the result of concatenating the computations $\Omega^1, \Omega^2, \ldots \Omega^m$ and Ω^0. Obviously, Γ is a computation such that

$$\Gamma \Vdash \langle d_\imath, \langle a_1, \ldots a_n \rangle, v_\imath \rangle \qquad (\text{for } \imath = 1, \ldots, m)$$

and

$$\Gamma \Vdash \langle d_0, \langle v_1, \ldots v_m \rangle, b \rangle \ .$$

Now, by Clause (R4) of Definition 7.5.1, we can conclude

$$\Gamma, \langle \langle 3, n, d_1, \ldots d_m, d_0 \rangle, \langle a_1, \ldots a_n \rangle, b \rangle$$

is a computation, moreover,

$$\Gamma, \langle \langle 3, n, d_1, \ldots d_m, d_0 \rangle, \langle a_1, \ldots a_n \rangle, b \rangle \Vdash \langle \langle 3, n, d_1, \ldots d_m, d_0 \rangle, \langle a_1, \ldots a_n \rangle, b \rangle \ .$$

In order to prove the converse implication, assume that

$$\Gamma \Vdash \langle \langle 3, n, d_1, \ldots d_m, d_0 \rangle, \langle a_1, \ldots a_n \rangle, b \rangle \ .$$

Then, the number $\langle \langle 3, n, d_1, \ldots d_m, d_0 \rangle, \langle a_1, \ldots a_n \rangle, b \rangle$ occurs in the computation Γ. By inspecting the definition of a computation, we can see that we have $v_1, \ldots, v_m \in \mathbb{N}$ such that

$$\Gamma \Vdash \langle d_\imath, \langle a_1, \ldots a_n \rangle, v_\imath \rangle \qquad (\text{for } \imath = 1, \ldots, m)$$

and

$$\Gamma \Vdash \langle d_0, \langle v_1, \ldots v_m \rangle, b \rangle \ .$$

(If this is not the case, then the number $\langle \langle 3, n, d_1, \ldots d_m, d_0 \rangle, \langle a_1, \ldots a_n \rangle, b \rangle$ cannot occur in Γ.) By our induction hypothesis, we have

$$g(a_1, \ldots, a_n) = v_\imath \qquad (\text{for } \imath = 1, \ldots, m)$$

and $h(v_1, \ldots, v_m) = b$. Hence, $f(a_1, \ldots, a_n) = b$.

There are two more inductive cases to deal with: The case when $e = \langle 4, n, d_0, d_1 \rangle$, and the case when $e = \langle 5, n, d \rangle$. The proofs for these two cases are similar to the proof for the case when $e = \langle 3, n, d_1, \ldots d_m, d_0 \rangle$, and we omit the details. □

> *Chaff:* Ha! You didn't really believe us back on page 225 when we said that we weren't going to make you fill in parts of the arguments, did you?

The next lemma is a straightforward consequence of the previous one. Just note that if Γ is a computation and γ is a triplet such that $\Gamma \Vdash \gamma$, then there exists an initial sequence of Γ where γ is the last triplet. This sequence is a computation.

Lemma 7.5.3. *Let e be a computable index for the function f. Then*

$$f(a_1, \ldots, a_n) = b \quad \Leftrightarrow$$
$$\text{there exists a computation of the form } \Gamma, \langle e, \langle a_1, \ldots, a_n \rangle, b \rangle \ .$$

(And thus, $f(a_1, \ldots, a_n)$ is defined if and only if there exists a computation of the form $\Gamma, \langle e, \langle a_1, \ldots, a_n \rangle, b \rangle$.)

In order to make our formulas more readable and make them fit the page, we need to develop some notation: We will, e.g., write $(\exists x, y \leq t)$ in place of $(\exists x \leq t)(\exists y \leq t)$. Furthermore, we will write $\exists_{\text{something}}[\ldots]$ and $\forall_{\text{something}}[\ldots]$ in place of, respectively, (∃something)[...] and (∀something)[...]. We will need the primitive recursive coding function introduced in Section 7.3. Recall that $(\langle x_1, \ldots, x_n \rangle)_i = x_i$ (for $i = 1, \ldots n$) and that $|\langle x_1, \ldots, x_n \rangle| = n$. We will write $(t)_{i_1, i_2, \ldots, i_n}$ in place of $(\ldots((t)_{i_1})_{i_2}\ldots)_{i_n}$. Do not confuse x_i and $(x)_i$. Remember that x_i denotes an indexed variable, while $(x)_i$ denotes the i^{th} element in the sequence encoded by x.

Lemma 7.5.4. *There exists a primitive recursive predicate C such that $C(t)$ holds if and only if $t = \langle c_1, c_2, \ldots, c_k \rangle$ and $k \geq 1$ and the sequence $(c_1, c_2, \ldots, c_k)$ is a computation.*

Proof. For each clause (Rj) of Definition 7.5.1, we will give a primitive recursive predicate P_j. Given that $((t)_1, (t)_2, \ldots, (t)_{i-1})$ is a computation, the predicate $P_j(t, i)$ holds if and only if the clause (Rj) allows us extend the computation $((t)_1, , (t)_2, \ldots, (t)_{i-1}$ by $(t)_i)$. The predicate $C(t)$ in our lemma holds if, and only if

$$\text{code}(t) \ \wedge \ |t| \geq 1 \ \wedge \ \forall_{i<|t|} [\, P_1(t, i+1) \ \vee \ P_2(t, i+1) \ \vee \ P_3(t, i+1) \ \vee \ P_4(t, i+1) \ \vee \ P_5(t, i+1) \ \vee \ P_6(t, i+1) \,] \ .$$

By the lemmas in Section 7.3, C is a primitive recursive predicate when P_1, P_2, P_3, P_4, P_5 and P_6 are primitive recursive predicates.

Let $P_1(t, i)$ be the predicate

$$\exists_{e,x,y \leq t} [\, (t)_i = \langle e, x, y \rangle \ \wedge \ e = \langle 0 \rangle \ \wedge \ |x| = 1 \ \wedge \ y = (x)_1 + 1 \,] \ .$$

Then, $P_1(t, i)$ states that t encodes a sequence of the form

$$(c_1, c_2, \ldots, c_{i-1}, \ \langle \ \langle 0 \rangle \ , \ \langle x_1 \rangle \ , \ x_1 + 1 \ \rangle \ , c_{i+1}, \ldots) \ .$$

Let $P_2(t, i)$ be the predicate

$$\exists_{e,x,y \leq t} [\, (t)_i = \langle e, x, y \rangle \ \wedge \ |e| = 3 \ \wedge \ (e)_1 = 1 \ \wedge \ |x| = (e)_2 \ \wedge \ 1 \leq (e)_3 \ \wedge \ (e)_3 \leq (e)_2 \ \wedge \ y = (x)_{(e)_3} \,] \ .$$

Then, $P_2(t, i)$ states that t encodes a sequence of the form

$$(c_1, c_2, \ldots, c_{i-1}, \ \langle \ \langle 1, n, \ell \rangle \ , \ \langle x_1, \ldots, x_n \rangle \ , \ x_\ell \ \rangle \ , c_{i+1}, \ldots)$$

where $1 \leq \ell \leq n$.

Let $P_3(t,i)$ be the predicate

$$\exists_{e,x \leq t} \left[(t)_i = \langle e, x, 0 \rangle \ \wedge \ e = \langle 2 \rangle \ \wedge \ |x| = 0 \right].$$

Then, $P_3(t,i)$ states that t encodes a sequence of the form

$$(c_1, c_2, \ldots, c_{i-1}, \ \langle \ \langle 2 \rangle \ , \ \langle \rangle \ , \ 0 \ \rangle \ , c_{i+1}, \ldots) \ .$$

Let $P_4(t,i)$ be the predicate

$$\exists_{e,x,y \leq t} \Big[(t)_i = \langle e, x, y \rangle \ \wedge \ |e| \geq 3 \ \wedge \ (e)_1 = 3 \ \wedge \ (e)_2 = |x| \ \wedge$$
$$\exists_{v \leq t} \Big\{ |v| = |e| \ \dot{-} \ 3 \wedge \forall_{j < |e|} \left[3 \leq j \ \rightarrow \ \exists_{k<i} \left[(t)_{k+1} = \langle (e)_j, x, (v)_{j \dot{-} 2} \rangle \right] \right] \wedge$$
$$\exists_{k<i} \left[(t)_{k+1} = \langle (e)_{|e|}, v, y \rangle \right] \Big\} \Big] \ .$$

Then, $P_4(t,i)$ states that t encodes a sequence of the form

$$(c_1, c_2, \ldots, c_{i-1}, \ \langle \ \langle 3, n, d_3, \ldots, d_k \rangle \ , \ \langle x_1, \ldots, x_n \rangle \ , \ y \ \rangle \ , c_{i+1}, \ldots)$$

where the sequence $(c_1, c_2, \ldots, c_{i-1})$ contains elements of the forms

$$\begin{array}{l} \langle \ d_3 \ , \ \langle x_1, \ldots, x_n \rangle \ , \ v_1 \ \rangle \\ \langle \ d_4 \ , \ \langle x_1, \ldots, x_n \rangle \ , \ v_2 \ \rangle \\ \vdots \\ \langle \ d_{k-1} \ , \ \langle x_1, \ldots, x_n \rangle \ , \ v_{k-3} \ \rangle \\ \langle \ d_k \ , \ \langle v_1, \ldots, v_{k-3} \rangle \ , \ y \ \rangle \end{array}$$

for some $v_1, \ldots v_{k-3}$.

Recall that

$$\langle x_1, \ldots x_n \rangle ^\frown \langle y_1, \ldots y_m \rangle \ = \ \langle x_1, \ldots x_n, y_1, \ldots y_m \rangle \ .$$

Let $\mathrm{clip}(x) = (\mu z \leq x)[z ^\frown (x)_{|x|} = x]$. Then, we have

$$\mathrm{clip}(\langle x_1, \ldots, x_n, y \rangle) ^\frown \langle z \rangle \ = \ \langle x_1, \ldots, x_n, z \rangle \ .$$

Now, let $P_5(t,i)$ be the predicate

$$\exists_{e,x,y \leq t} \Big[(t)_i = \langle e, x, y \rangle \ \wedge \ |e| = 4 \ \wedge \ (e)_1 = 4 \ \wedge \ (e)_2 + 1 = |x| \ \wedge$$
$$\exists_{v \leq t} \Big\{ |v| = (x)_{|x|} + 1 \ \wedge \ \exists_{k<i} \left[(t)_{k+1} = \langle \ (e)_3 \ , \ \mathrm{clip}(x) \ , \ (v)_1 \ \rangle \right] \wedge$$
$$\forall_{j < (x)_{|x|}} \exists_{k<i} \left[(t)_{k+1} = \langle \ (e)_4 \ , \ \mathrm{clip}(x) ^\frown \langle j, (v)_{j+1} \rangle \ , \ (v)_{j+2} \ \rangle \right] \wedge y = (v)_{|v|} \ \Big\} \Big] \ .$$

The predicate $P_5(t,i)$ states that t encodes a sequence of the form

$$(c_1, c_2, \ldots, c_{i-1}, \ \langle \ \langle 4, n, a, b \rangle \ , \ \langle x_1, \ldots, x_n, \ell \rangle \ , \ y \ \rangle \ , c_{i+1}, \ldots)$$

where the sequence $(c_1, c_2, \ldots, c_{i-1})$ contains elements of the forms

$$\begin{array}{l} \langle\, a\ ,\ \langle x_1, \ldots, x_n\rangle\ ,\ v_1\,\rangle \\ \langle\, b\ ,\ \langle x_1, \ldots, x_n, 0, v_1\rangle\ ,\ v_2\,\rangle \\ \langle\, b\ ,\ \langle x_1, \ldots, x_n, 1, v_2\rangle\ ,\ v_3\,\rangle \\ \vdots \\ \langle\, b\ ,\ \langle x_1, \ldots, x_n, \ell-2, v_{\ell-1}\rangle\ ,\ v_\ell\,\rangle \\ \langle\, b\ ,\ \langle x_1, \ldots, x_n, \ell-1, v_\ell\rangle\ ,\ y\,\rangle \end{array}$$

for some $v_1, \ldots v_\ell$.

Let $P_6(t, i)$ be the predicate

$$\exists_{e,x,y\le t}\Big[\,(t)_i = \langle e, x, y\rangle\ \wedge\ |e| = 3\ \wedge\ (e)_1 = 5\ \wedge\ (e)_2 = |x|\ \wedge \\ \forall_{j<y}\exists_{k<i}\exists_{w\le t}\,[\,(t)_{k+1} = \big\langle\,(e)_3\,,\,x^\frown\langle j\rangle\,,\,w\,\big\rangle\ \wedge\ w \neq 0\,]\ \wedge \\ \exists_{k<i}\,[\,(t)_{k+1} = \big\langle\,(e)_3\,,\,x^\frown\langle y\rangle\,,\,0\,\big\rangle\,]\,\Big]\ .$$

Then, $P_6(t, i)$ states that t encodes a sequence of the form

$$(c_1, c_2, \ldots, c_{i-1},\ \langle\,\langle 5, n, a\rangle\ ,\ \langle x_1, \ldots, x_n\rangle\ ,\ y\,\rangle\,, c_{i+1}, \ldots)$$

where the sequence $(c_1, c_2, \ldots, c_{i-1})$ contains elements of the forms

$$\begin{array}{l} \langle\, a\ ,\ \langle x_1, \ldots, x_n, 0\rangle\ ,\ v_0\,\rangle \\ \langle\, a\ ,\ \langle x_1, \ldots, x_n, 1\rangle\ ,\ v_1\,\rangle \\ \vdots \\ \langle\, a\ ,\ \langle x_1, \ldots, x_n, y-1\rangle\ ,\ v_{y-1}\,\rangle \\ \langle\, a\ ,\ \langle x_1, \ldots, x_n, y\rangle\ ,\ 0\,\rangle \end{array}$$

for some $v_1, \ldots v_{y-1}$ such that $v_i \neq 0$ (for $i = 0 \ldots y-1$).

Assume that $((t)_1, (t)_2, \ldots, (t)_{i-1})$ is a computation. Then, it is a tedious, but manageable, job to check that $P_j(t, i)$ holds if and only if Clause (Rj) of Definition 7.5.1 says that $((t)_1, (t)_2, \ldots, (t)_i)$ is a computation. The predicates $P_1, \ldots, P_6$ are primitive recursive by the lemmas in Section 7.3. □

We are ready to give the definition of Kleene's T-predicate and complete our proof of Kleene's Normal Form Theorem.

Definition 7.5.5. For each $n \in \mathbb{N}$, we define *Kleene's T-predicate* $\mathcal{T}_n$ by

$$\mathcal{T}_n(e, x_1, \ldots x_n, t) \quad \Leftrightarrow \\ C(t)\ \wedge\ (t)_{|t|,1} = e\ \wedge\ |(t)_{|t|,2}| = n\ \wedge\ (t)_{|t|,2,1} = x_1\ \wedge\ \ldots\ \wedge\ (t)_{|t|,2,n} = x_n$$

where C is the predicate from Lemma 7.5.4. (Note that $(t)_{|t|}$ denotes the last element in the sequence encoded by t.) Furthermore, we define the function $\mathcal{U}$ by $\mathcal{U}(t) = (t)_{|t|,3}$.

The predicate $\mathcal{T}_n$ is primitive recursive by the lemmas in Section 7.3. Moreover, $\mathcal{T}_n(e, x_1, \ldots x_n, t)$ holds if and only if $t = \langle t_1, \ldots, t_k \rangle$, where $(t_1, \ldots, t_k)$ is a computation and t_k is of the form $\langle e, \langle x_1, \ldots, x_n \rangle, y \rangle$. Let e be an index for f. By Lemma 7.5.3, we have

$$f(x_1, \ldots x_n) \text{ is defined } \Leftrightarrow$$
$$\text{there exists } t \text{ such that } \mathcal{T}_n(e, x_1, \ldots x_n, t) \text{ holds .}$$

The function $\mathcal{U}$ is primitive recursive by the lemmas in Section 7.3. Let e be an index for f, and assume that $f(x_1, \ldots x_n)$ is defined. By Lemma 7.5.3, we have

$$f(x_1, \ldots x_n) \;=\; \mathcal{U}((\mu t)[\mathcal{T}_n(e, x_1, \ldots x_n, t)]) \; .$$

It is easy to see that $\mathcal{T}_n(e, x_1, \ldots x_n, t)$ does not hold (for any $x_1, \ldots x_n, t \in \mathbb{N}$) if e is not an index for an n-ary function.

This (finally) completes the proof of Theorem 7.4.3.

7.5.1 Exercises

1. We define the function h by

$$h(x, y, z) = \mathcal{S}(\mathcal{I}_3^3(x, y, z))$$

and we define the function f by

$$\begin{aligned} f(x, 0) &= \mathcal{I}_1^1(x) \\ f(x, y+1) &= h(x, y, f(x, y)) \end{aligned}$$

Give a computation which shows that $f(2, 2) = 4$. Proceed the following way: Find a computable index e for f (see Exercise 1 of Section 7.4.1). Then, find a computation Γ such that $\Gamma \Vdash \langle e, \langle 2, 2 \rangle, 4 \rangle$. Finally, use Lemma 7.5.2 to conclude that $f(2, 2) = 4$.

2. Give the details in the proof of Lemma 7.5.2 for the case when $e = \langle 4, n, d_0, d_1 \rangle$.

3. In this exercise you should proceed as in the proof of Lemma 7.5.4: Write up formulas that define the predicates, and use the Lemmas in Section 7.3 to conclude that the predicates are primitive recursive.

   A natural number t encodes a finite sequence $(c_1, \ldots, c_n)$ of natural numbers when $t = \langle c_1, \ldots, c_n \rangle$ (we have $(t)_i = c_i$ for $i = 1 \ldots n$).

   (a) We say that a finite sequence $(c_1, c_2, \ldots, c_n)$ of natural numbers is a *Fibonacci sequence* if

   - $n \geq 2$,

- $c_1 = c_2 = 1$, and
- for $i = 1 \ldots, n-2$, we have $c_{i+2} = c_{i+1} + c_i$.

Let $P(t)$ be the predicate that holds iff t encodes a Fibonacci sequence. Prove that P is a primitive recursive predicate.

(b) We say that a finite sequence $(c_1, c_2, \ldots, c_n)$ of natural numbers is a *generalized Fibonacci sequence* if

- $n \geq 2$,
- $c_1 = c_2 = 1$, and
- for $i = 3, \ldots, n$, we have $c_i = c_j + c_k$ where j and k are two different numbers strictly less than i.

Let $P(t)$ be the predicate that holds iff t encodes a generalized Fibonacci sequence. Prove that P is a primitive recursive predicate.

(c) We say that a finite sequence $c_1, c_2, \ldots, c_n$ of natural numbers is a *generalized generalized Fibonacci sequence* if

- $n \geq 2$ and $c_1 = c_2 = 1$
- for $i = 3, \ldots n$, there exists k such that $1 \leq k < i$ and $c_i = c_{j_1} + c_{j_2} + \ldots + c_{j_k}$ where $j_1, \ldots j_k$ are k different numbers strictly less than i.

Let $P(t)$ be the predicate that holds iff t encodes a generalized generalized Fibonacci sequence. Prove that P is a primitive recursive predicate.

4. We define the sequence $(c_0, c_1, c_2, \ldots)$ by $c_0 = 1$ and $c_{i+1} = \sum_{j \leq i} c_j$. A natural number t encodes the finite sequence $(c_0, \ldots, c_n)$ when $t = \langle c_0, \ldots, c_n \rangle$ (we have $(t)_{i+1} = c_i$ for $i = 0 \ldots n$).

(a) Let $P(t, n)$ be the predicate that holds iff t encodes $(c_0, \ldots, c_n)$. Prove that P is a primitive recursive predicate.
Let $f(i) = c_i$. (Thus, $f(n+1) = f(n) + f(n-1) + \ldots + f(0)$.)

(b) Use the predicate in (a) to prove that f is a computable function.

(c) Use the predicate in (a) to prove that f is a primitive recursive function.

5. (a) Assume that $\{e\}^n(x_1, \ldots, x_n) = m$, and assume that the predicate $\mathcal{T}_n(e, x_1, \ldots, x_n, t)$ holds. Explain why we have $m < t$.

(b) Prove that there exists a computable function f such that

$$\{e\}^1(x) = m \;\; \Rightarrow \;\; m < f(\langle e, x \rangle) \; .$$

(c) Prove that there does not exist a *total* computable function f such that

$$\{e\}^1(x) = m \;\; \Rightarrow \;\; m < f(\langle e, x \rangle) \; .$$

(Hint: Prove that the function d from Corollary 7.4.7 is computable if such an f exists.)

6. Let A_n^k denote the k^{th} iterate of the n^{th} branch of the Ackermann function (see Exercise 9 in Section 7.3.1) Prove that a function f is primitive recursive if and only if there exist $e, k, n \in \mathbb{N}$ such that

$$f(x_1, \ldots, x_m) \;=\; \mathcal{U}(\,(\mu t \leq A_n^k(\max(x_1, \ldots, x_m))[\,\mathcal{T}_m(e, x_1, \ldots x_m, t)\,]\,)\,.$$

7.6 Semi-Computable and Computably Enumerable Sets

Recall the definition of a computable set: a set is computable if the characteristic function for the set is computable. A characteristic function is a total function. Hence, when a set is computable we have an algorithm for deciding membership in the set. The algorithm will always yield an answer – either YES or NO. That is why computable sets sometimes also are called *decidable sets*.

In this section we introduce the *semi-computable sets* and the *computably enumerable sets*:

- When a set S is semi-computable, there might not be an algorithm for *deciding* membership in S, but there will be an algorithm for *confirming* membership in S. The algorithm, let's call it $\mathcal{A}$, will confirm membership by terminating on input x if x is in S. If x is not in S, the algorithm $\mathcal{A}$ will never terminate on input x. Hence, we can confirm membership in S in finite time, but we cannot necessarily confirm that an element is not an element of S in finite time. If, for example, $17 \notin S$ and we start our algorithm $\mathcal{A}$ running with input 17, and we come back and check on our algorithm in 2057 and $\mathcal{A}$ is still humming along, we won't know if that means that $17 \notin S$ or if it means that $17 \in S$ and $\mathcal{A}$ will tell us that if we only wait another 1500 years.

- The computably enumerable sets are those sets whose members can be enumerated by an algorithm. When a set is computably enumerable, there is an algorithm for generating a list $a_0, a_1, a_2, \ldots$ such that every a_i appearing in the list will be a member of the set and, sooner or later, each member of the set will show up in the list.

Technically, we will define a set to be semi-decidable if it is the domain of a computable function; and we will define a set to be computably enumerable if it is the range of a total computable function. Any finite set will also be computably enumerable by definition.

Before we turn to our formal definitions, let us test out our intuitions. If there is an algorithm for listing the members in a set A, there will also be an algorithm for confirming that a is a member of A: List the members of A (we assume that A is infinite). If a shows up in list, the algorithm

terminates (and thus confirms that a is an element of A). If a never shows up, the algorithm will never terminate. It will continue to list elements of A forever and ever. Thus, our intuitions definitely tell us that any computably enumerable set also should be semi-decidable. But should any semi-decidable set be computably enumerable? If there is an algorithm for confirming membership in a set, will there also be an algorithm for listing the members of the set? Perhaps you want to think about this question a little bit before reading on.

Let $\texttt{confirm}_A$ be an algorithm that confirms membership in the set A, that is, $\texttt{confirm}_A$ terminates on input a if $a \in A$, and $\texttt{confirm}_A$ does not terminate on input a if $a \notin A$. Some way or another we count the number of steps in an execution of $\texttt{confirm}_A$. Any reasonable definition of a *step* will do. Now, pick a number that is in A (we assume that A is nonempty). Let us say that 17 is the number we pick. For an arbitrary pair x, y of natural numbers, we can carry out the following procedure:

- execute $\texttt{confirm}_A$ on input x
- count the number of steps in this execution of $\texttt{confirm}_A$
- if $\texttt{confirm}_A$ terminates before y steps is executed, add the number x to a list L (we have confirmed that x is in A)
- if $\texttt{confirm}_A$ does not terminate before y steps is executed, add the number 17 to a list L (we know that 17 is in A).

Any number that this procedure adds to the list L will certainly be in A. Moreover, if we start with an empty list L and execute this procedure over and over again – one time for each possible pair (x, y) of natural numbers – then every member of A will sooner or later be added to the list L. Thus, intuitively, if there is an algorithm for confirming membership in a set, there is also an algorithm for listing the members of the set. And if there is an algorithm that lists the members of a set, then there is of course also an algorithm that lists each member of the set only once: It is straightforward to avoid adding an element to a list twice. Any semi-computable set should be computably enumerable.

Our formalized mathematical theory will of course be in accordance with our intuitions about semi-computable and computably enumerable sets. We will prove formally that a set is semi-computable if and only if it is computably enumerable. We will also prove that there exists semi-computable (and thus also computable enumerable) sets that are not computable.

A few words on terminology: We will talk about computable sets, and we will talk about decidable sets. They are the same thing. So a decidable set is nothing but a computable set and a computable set is nothing but a decidable set. Moreover, semi-decidable means the same as semi-computable. But we will tend to use the word "decidable" in our informal discussions, and we will tend to use the word "computable" in more formal

passages like proofs and definitions. Please, also remember that what we may refer to as problems, languages, relations, predicates, and so on, easily can be reduced to sets. So even if the notions of being computable, semi-computable, and computably enumerable just are formally defined for sets, it makes perfectly good sense to say that problems, languages, relations, predicates, and so on, are computable, semi-computable and computably enumerable. In the days when computability theory was called recursion theory, computable sets were called *recursive sets*, and computably enumerable sets were called *recursively enumerable sets*. Moreover, *recursively enumerable* was abbreviated *r.e.* Hence, an r.e. set is nothing but a computably enumerable set. Many authors do still stick to the old terminology and prefer to talk about recursive and recursively enumerable sets.

As we begin our formal definitions, let us first agree on some notation and abbreviations that are designed to improve the readability of what is to follow. From now on we will use $\mathcal{T}$ to denote Kleene's T-predicate $\mathcal{T}_1$. We will also write $\{e\}$ in place of $\{e\}^1$ (thus, $\{e\}(x) = \mathcal{U}((\mu t)[\mathcal{T}(e, x, t)])$). Furthermore, we will use $\mathrm{dom}(f)$ and $\mathrm{rng}(f)$ to denote, respectively, the domain and the range of the unary computable function f, that is,

$$\mathrm{dom}(f) \;=\; \{x \mid \text{there exists } y \in \mathbb{N} \text{ such that } f(x) = y\}$$

and

$$\mathrm{rng}(f) \;=\; \{y \mid \text{there exists } x \in \mathbb{N} \text{ such that } f(x) = y\}\,.$$

Definition 7.6.1. A set A of natural numbers is *semi-computable* if there exists a computable function f such that $A = \mathrm{dom}(f)$.

Definition 7.6.2. A set A of natural numbers is *computably enumerable* if

- A is finite, or
- there exists a total computable one-to-one function f such that $A = \mathrm{rng}(f)$.

The proof of the next lemma is an exercise for the reader.

Lemma 7.6.3. *If e is a computable index for f, then*

$$\mathrm{dom}(f) \;=\; \{x \mid \textit{there exists } t \textit{ such that } \mathcal{T}(e, x, t)\}\,.$$

Theorem 7.6.4. *Any computable set is semi-computable.*

Proof. Let A be a computable set. By Definition 7.2.4, the characteristic function χ_A is computable. Let $f(x) = (\mu y)[y = y \wedge \chi_A(x) = 0]$. The function f is computable as the computable functions are closed under minimalization. Moreover, $\mathrm{dom}(f) = A$. Hence, A is semi-computable by Definition 7.6.1. □

Theorem 7.6.5. *Let A be a set of natural numbers. The following assertions are equivalent:*

(1) A is a semi-computable set

(2) $A = \emptyset$, or A is the range of a primitive recursive (and thus, a total and computable) function

(3) A is a computably enumerable set.

Proof. First we prove that (1) implies (2). Assume (1). By Definition 7.6.1, we have a computable function f such that $A = \mathrm{dom}(f)$. Let e be a computable index for f. By Lemma 7.6.3, we have

$$A \;=\; \mathrm{dom}(f) \;=\; \{x \mid \text{there exists } t \text{ such that } \mathcal{T}(e, x, t)\} \qquad (*)$$

Furthermore, assume that $A \neq \emptyset$, and let a be an element of A. We can without loss of generality assume that a is 17. Let

$$g(x) \;=\; \begin{cases} (x)_1 & \text{if } \mathcal{T}(e, (x)_1, (x)_2) \\ 17 & \text{otherwise} \end{cases} \qquad (**)$$

where $(x)_i$ is the decoding function from Section 7.3. The Kleene T-predicate $\mathcal{T}$ is primitive recursive, as are the decoding functions. Moreover, the class of primitive recursive functions is closed under composition and definition by cases. Hence, g is a primitive recursive function. If we can show that $\mathrm{rng}(g) = A$, we will have established (2).

We first prove that $A \subseteq \mathrm{rng}(g)$. Assume $b \in A$. By (*), there exists t such that the predicate $\mathcal{T}(e, b, t)$ holds. Then, by (**), we have

$$g(\langle b, t\rangle) \;=\; \begin{cases} b & \text{if } \mathcal{T}(e, b, t) \\ 17 & \text{otherwise} \end{cases} \;=\; b$$

and we conclude that $b \in \mathrm{rng}(g)$. This proves that $A \subseteq \mathrm{rng}(g)$.

We now prove that $\mathrm{rng}(g) \subseteq A$. Assume that $b \in \mathrm{rng}(g)$. Now, if b is 17 then we obviously have $17 \in A$ (17 is an element that we have chosen from A). If b is different from 17, then it follows from (**) that there has to be a t such that the predicate $\mathcal{T}(e, b, t)$ holds. When such a t exists, we have $b \in A$ by (*). This proves that $\mathrm{rng}(g) \subseteq A$, which establishes (2), as needed.

It requires some technical work to prove that (2) implies (3). If A is finite, it follows trivially from our definitions that the implication holds. Now, assume that (2) holds and that A is infinite. Then, we have a primitive recursive function f such that $A = \mathrm{rng}(f)$. (We will provide a total computable one-to-one function g such that $A = \mathrm{rng}(g)$.) We define the function $\hat{f}$ by

$$\hat{f}(x) = \begin{cases} 1 & \text{if } (\forall j \leq x)[f(j) \neq f(x+1)] \\ 0 & \text{otherwise.} \end{cases}$$

The function $\hat{f}$ is primitive recursive by the lemmas of Section 7.3. Furthermore, if $\hat{f}(n) = 1$, the value $f(n+1)$ does not occur amongst the values in the list $f(0), f(1), \ldots, f(n)$, and if is $\hat{f}(n) = 0$, the value $f(n+1)$ occurs amongst the values in the list $f(0), f(1), \ldots, f(n)$. Hence, the number of different values in the list $f(0), f(1), \ldots, f(n)$ will be $(\sum_{i \leq n} \hat{f}(i)) + 1$. Now, let

$$g(x) = \begin{cases} f(0) & \text{if } x = 0 \\ f\left((\mu i)\left[(\sum_{j \leq i} \hat{f}(j)) = x\right]\right) & \text{otherwise.} \end{cases}$$

Then, g is a total computable one-to-one function such that $\mathrm{rng}(g) = A$. By Definition 7.6.2, A is a computably enumerable set, which proves that (2) implies (3).

> *Chaff:* Did you notice that we used unbounded minimalization to define the function g? It is *not* true that every infinite computably enumerable set is the range of primitive recursive one-to-one function, so the use of that unbounded minimalization was necessary.)

To complete the proof of the theorem, we must show that (3) implies (1), but that is not difficult. Assume that A is computably enumerable. If A is finite, let us say that $A = \{17, 714, 9999\}$, let

$$f(x) = (\mu i)[(i = 17 \ \vee \ i = 714 \ \vee \ i = 9999) \ \wedge \ i = x]$$

and f is a computable function such that $A = \mathrm{dom}(f)$. If A is infinite and $A = \mathrm{rng}(g)$ where g is a total computable function, let $f(x) = (\mu i)[g(i) = x]$. Then, f is a computable function such that $A = \mathrm{dom}(f)$. Thus, (3) implies (1). □

Now, if we have an algorithm for enumerating the members of a set A, and we also have an algorithm for enumerating the members of the complementary set $\overline{A}$, what is the situation then? Well, intuitively, we should then have an algorithm for deciding membership in A: We can generate two lists simultaneously. One list ℓ containing elements of A, and another list $\overline{\ell}$ containing elements of $\overline{A}$. An input a to this algorithm will show up in one, and only one, of the lists. If a shows up in ℓ, we can conclude that a is in A. If a shows up in $\overline{\ell}$, we can conclude that a is not in A.

So, if A and $\overline{A}$ are computably enumerable sets, then A should be a computable set. That is what our intuition about algorithms tells us. The next theorem shows that our formalized mathematical theory mirrors our intuition.

Theorem 7.6.6. *A set of natural numbers A is computable if and only if both A and its complement $\overline{A}$ are computably enumerable.*

Proof. Assume A is computable. Then, $\overline{A}$ is also computable (see Exercise 5). By Theorem 7.6.4 and Theorem 7.6.5, both A and $\overline{A}$ are computably enumerable sets.

Now, assume that the two sets A and $\overline{A}$ are computably enumerable. If one of the sets is empty, then it is obvious that A is computable. Thus, assume that neither A nor $\overline{A}$ are empty. By Theorem 7.6.5, we have primitive recursive functions f_0 and f_1 such that $A = \text{rng}(f_0)$ and $\overline{A} = \text{rng}(f_1)$. Let $f(x) = (\mu i)[f_0(i) = x \vee f_1(i) = x]$. Observe that f is a total computable function. Now, let

$$\chi(x) = \begin{cases} 0 & \text{if } f_0(f(x)) = x \\ 1 & \text{otherwise.} \end{cases}$$

Then, χ is a computable function. Moreover, χ is the characteristic function for the set A. Thus, A is a computable set. □

The next corollary follows straightforwardly from the two previous theorems.

Corollary 7.6.7. *A set of natural numbers A is computable if and only if both A and its complement $\overline{A}$ are semi-computable.*

We have enumerated the unary computable functions:

$$\{0\}, \{1\}, \{2\}, \{3\}, \ldots.$$

Since a semi-computable set is defined as the domain of a unary computable function, we have an enumeration of the semi-computable sets for free. Let $\mathcal{W}_0$ be the domain of the function $\{0\}$, let $\mathcal{W}_1$ be the domain of the function $\{1\}$, and so on. Then, a set is semi-computable if and only if it appears in the sequence $\mathcal{W}_0, \mathcal{W}_1, \mathcal{W}_2, \mathcal{W}_3, \ldots$.

Definition 7.6.8. We define the e^{th} *semi-computable set*, written $\mathcal{W}_e$, by $\mathcal{W}_e = \text{dom}(\{e\})$.

Lemma 7.6.9. *A set of natural numbers A is semi-computable if and only if there exists e such that $A = \mathcal{W}_e$.*

Proof. We know that

$$A \text{ is semi-computable} \Leftrightarrow \text{there exists a computable function } f \text{ such that } A = \text{dom}(f) \quad (1)$$

by Definition 7.6.1. We also know that

$$A \text{ is semi-computable} \Leftrightarrow \text{there exists } e \in \mathbb{N} \text{ such that } A = \text{dom}(\{e\}) \quad (2)$$

by (1) and the Enumeration Theorem 7.4.5. Finally, we have

$$A \text{ is semi-computable} \quad \Leftrightarrow \quad \text{there exists } e \in \mathbb{N} \text{ such that } A = \mathcal{W}_e$$

by (2) and Definition 7.6.8. □

Definition 7.6.10. We define the set $\mathcal{K}$ by $\mathcal{K} = \{x \mid x \in \mathcal{W}_x\}$.

The set $\mathcal{K}$ will play a central role throughout the rest of this chapter. We will use properties of this set to give a smooth proof of Gödel's First Incompleteness Theorem. Our proof of the undecidability of the Entscheidungsproblem will be also based on properties of $\mathcal{K}$.

Before studying the proofs of the next few theorems, the reader should note that $0 \in \mathcal{K}$ iff $0 \in \mathcal{W}_0$; that $1 \in \mathcal{K}$ iff $1 \in \mathcal{W}_1$; and so on. Thus, $\mathcal{K}$ is semi-computable since $\mathcal{K}$ is the domain of the computable function f given by $f(x) = \{x\}(x)$. Moreover, $0 \in \overline{\mathcal{K}}$ iff $0 \notin \mathcal{W}_0$; and $1 \in \overline{\mathcal{K}}$ iff $1 \notin \mathcal{W}_1$; and so on. Thus, by an easy diagonalization argument we can conclude that the set $\overline{\mathcal{K}}$ cannot be semi-computable! The set cannot be semi-computable because it cannot appear in the sequence $\mathcal{W}_0, \mathcal{W}_1, \mathcal{W}_2, \mathcal{W}_3, \ldots$. The set $\overline{\mathcal{K}}$ cannot be $\mathcal{W}_0$ as $0 \notin \overline{\mathcal{K}}$ iff $0 \in \mathcal{W}_0$; the set $\overline{\mathcal{K}}$ cannot be $\mathcal{W}_1$ as $1 \notin \overline{\mathcal{K}}$ iff $1 \in \mathcal{W}_1$; the set $\overline{\mathcal{K}}$ cannot be $\mathcal{W}_2 \ldots$; and so on. So we conclude that $\overline{\mathcal{K}}$ is not a semi-computable set, and then we can also conclude that $\mathcal{K}$ is not a computable set. If $\mathcal{K}$ were computable, then its complement set $\overline{\mathcal{K}}$ would be semi-computable. Let us state and prove this more formally:

Theorem 7.6.11. *The set $\overline{\mathcal{K}}$ is not semi-computable.*

Proof. Assume for the sake of a contradiction that $\overline{\mathcal{K}}$ is a semi-computable set. By Lemma 7.6.9, there exists m such that $\overline{\mathcal{K}} = \mathcal{W}_m$. Now, we have

$$m \in \overline{\mathcal{K}} \quad \Leftrightarrow \quad m \in \mathcal{W}_m \quad \Leftrightarrow \quad m \in \mathcal{K} \quad \Leftrightarrow \quad m \notin \overline{\mathcal{K}}\,.$$

The first equivalence holds since $\overline{\mathcal{K}} = \mathcal{W}_m$, the second equivalence holds by the definition of $\mathcal{K}$, and the third equivalence holds trivially. It is of course a contradiction that a number is in a set if and only if it is not in the set. □

Theorem 7.6.12. *The set $\mathcal{K}$ is semi-computable, but not computable.*

Proof. First we prove that $\mathcal{K}$ is semi-computable. We have

$$\begin{array}{lcll} x \in \mathcal{K} & \Leftrightarrow & x \in \mathcal{W}_x & \text{(definition of } \mathcal{K}) \\ & \Leftrightarrow & x \in \mathrm{dom}(\{x\}) & \text{(definition of } \mathcal{W}_x) \\ & \Leftrightarrow & \text{exists } t \text{ such that } \mathcal{T}(x,x,t). & \text{Lemma 7.6.3} \end{array}$$

Let $f(x) = (\mu i)[\mathcal{T}(x,x,i)]$. Then f is a computable function and $\mathrm{dom}(f) = \mathcal{K}$. By Definition 7.6.1, $\mathcal{K}$ is a semi-computable set.

Corollary 7.6.7 states that a set A is computable if and only if both A and $\overline{A}$ are semi-computable. Theorem 7.6.11 states that $\overline{\mathcal{K}}$ is not semi-computable. It follows that $\mathcal{K}$ is not computable. □

For an alternative proof of the previous theorem, see Exercise 9.

So $\mathcal{K}$ is not a computable set. An algorithm which terminates for any input a and answers the question, "Is a an element of $\mathcal{K}$?" correctly, has something in common with the King of France: It does not exist. Still, we do have some computational control over $\mathcal{K}$ as the set is semi-decidable and, thus, also computably enumerable. An algorithm can confirm membership in $\mathcal{K}$. An algorithm can list the elements of $\mathcal{K}$. The complement of $\mathcal{K}$, the set $\overline{\mathcal{K}}$, is not even semi-decidable. Any attempt to construct an algorithm confirming membership $\overline{\mathcal{K}}$, is bound to fail. There exist algorithms that can semi-decide infinite subsets of $\overline{\mathcal{K}}$, but these subsets will be *strict* subsets of $\overline{\mathcal{K}}$. Thus, if we know that the semi-decidable set $\mathcal{W}_e$ is a subset of $\overline{\mathcal{K}}$, we can conclude that there is a number in $\overline{\mathcal{K}}$ that is not in $\mathcal{W}_e$. There has to be such a number, otherwise $\overline{\mathcal{K}}$ would be a semi-decidable set. Moreover, when $\mathcal{W}_e$ is a subset of $\overline{\mathcal{K}}$, the index e itself is such a number. The proof of the next lemma explains why.

Lemma 7.6.13. *For any $e \in \mathbb{N}$, we have*

$$\mathcal{W}_e \subseteq \overline{\mathcal{K}} \;\Rightarrow\; e \in \overline{\mathcal{K}} \setminus \mathcal{W}_e \,.$$

(Recall that, for any sets A and B, $A \setminus B$ is the set $\{x \in A \mid x \notin B\}$.)

Proof. Assume that $\mathcal{W}_e \subseteq \overline{\mathcal{K}}$, and recall that $\overline{\mathcal{K}} = \{x \mid x \notin \mathcal{W}_x\}$. It follows from this assumption that

$$a \notin \mathcal{W}_a \text{ for any } a \in \mathcal{W}_e. \qquad (*)$$

We cannot have $e \in W_e$ since this contradicts (*). Thus, we conclude that $e \notin \mathcal{W}_e$. But then, as $\overline{\mathcal{K}} = \{x \mid x \notin \mathcal{W}_x\}$, we have $e \in \overline{\mathcal{K}}$. □

The proof of the preceding lemma is really not too bad, even if it might look a bit mysterious at first sight. The lemma is a straightforward consequence of the definition of $\overline{\mathcal{K}}$, but it is a consequence that will play a crucial role in our computability-theoretic proof of Gödel's First Incompleteness Theorem.

7.6.1 Exercises

1. Prove Lemma 7.6.3. Moreover, prove that

$$\mathrm{rng}(f) \;=\; \{y \mid \text{there exist } t, x \text{ such that } \mathcal{T}(e, x, t) \text{ and } \mathcal{U}(t) = y\}$$

where e is any computable index for f.

2. Let $A \subseteq \mathbb{N}$, and let P be a primitive recursive predicate such that

$$x \in A \quad \Leftrightarrow \quad \text{there exists } y \text{ such that } P(x, y) \text{ holds.}$$

Prove that A is a semi-computable set.

3. Let A be a semi-computable set. Prove that there exists a primitive recursive predicate P such that

$$x \in A \quad \Leftrightarrow \quad \text{there exists } y \text{ such that } P(x, y) \text{ holds.}$$

4. A total function $f : \mathbb{N} \to \mathbb{N}$ is non-decreasing when we have $f(x) \leq f(x + 1)$ (for any $x \in \mathbb{N}$). Let $A \subseteq \mathbb{N}$. The following assertions are equivalent:

   (1) A is computable

   (2) $A = \emptyset$, or A is the range of a non-decreasing total computable function

   (a) Prove that (1) implies (2).

   (b) Prove that (2) implies (1).

5. Prove that the class of computable sets is closed under complement, union, and intersection. (Assume that A and B are computable sets, and prove that $A \cup B$ and $A \cap B$ and $\overline{A}$ also are computable sets.)

6. We have seen that the semi-computable sets are not closed under complement, since the set $\mathcal{K}$ is semi-computable, but $\overline{K}$ is not. Prove that the class of semi-computable sets is closed under union and intersection. (Assume that A and B are semi-computable sets, and prove that $A \cup B$ and $A \cap B$ also are semi-computable sets.)

7. Prove that the semi-computable sets are not closed under set subtraction, that is, prove that there exist semi-computable sets A and B such that the set $A \setminus B$ is not semi-computable.

8. Give a direct proof of Corollary 7.6.7. Prove the corollary without using Theorem 7.6.5 and Theorem 7.6.6.

9. Recall how we proved that the set $\mathcal{K}$ is not computable:

   - first we proved that the complement set $\overline{\mathcal{K}}$ is not semi-computable (Theorem 7.6.11),
   - then, by Corollary 7.6.7, we concluded that $\mathcal{K}$ is not computable.

   It is possible to prove that $\mathcal{K}$ is not computable without first proving Corollary 7.6.7 (and Theorem 7.6.11 and Theorem 7.6.6). Try to find such a proof. Hint: Use a result from Section 7.4.

10. A set of natural numbers A *is m-reducible to* a set of natural numbers B, written $A \leq_m B$, if there exists a total computable function f such that

$$a \in A \quad \Leftrightarrow \quad f(a) \in B \; .$$

    (a) Explain why $\emptyset$ is the only set that is m-reducible to $\emptyset$.

    (b) Prove that $\leq_m$ is a reflexive and transitive relation.
    We say that a set A is nontrivial if $A \neq \emptyset$ and $A \neq \mathbb{N}$.

    (c) Let A be a computable set, and let B be any nontrivial set. Prove that $A \leq_m B$.

    (d) Let A be a computable set. Prove that $\mathcal{K} \not\leq_m A$.

    (e) Prove that any set m-reducible to $\mathcal{K}$ is a semi-computable set.

    (f) Let the set A be semi-computable and nontrivial. Prove that $A \leq_m \overline{A}$ if and only if A is computable.

11. The relation $\leq_m$ is defined in the previous exercise. We define the set $\mathcal{K}_0$ by

$$\mathcal{K}_0 \;=\; \{\, \langle x, y \rangle \mid x \in \mathcal{W}_y \,\} \; .$$

    (a) Let H be the predicate from Theorem 7.4.8 (Undecidability of the Halting Problem). Prove that $\langle a, b \rangle \in \mathcal{K}_0$ if and only if $H(b, a)$ holds.

    (b) Prove that any semi-computable set is m-reducible to $\mathcal{K}_0$.

    (c) Prove that $\mathcal{K}$ is m-reducible to $\mathcal{K}_0$.

    (d) Prove that $\mathcal{K}_0$ is m-reducible to $\mathcal{K}$ (this is difficult).

    (e) Prove that a set is semi-computable if and only if it is m-reducible to $\mathcal{K}$.

7.7 Applications to First-Order Logic

7.7.1 The Entscheidungsproblem

We have seen that Hilbert's Entscheidungsproblem was a substantial motivation for the development of computability theory. In this section we will prove that the Entscheidungsproblem is undecidable. Then, in the next section, we will prove another celebrated theorem with which you are already familiar: Gödel's First Incompleteness Theorem. Gödel proved this theorem a few years before Turing and Kleene developed computability theory, so he was not able to approach his result using the machinery that we have surveyed in this chapter. Indeed, before he became familiar with Turing's work, Gödel believed that it was impossible to capture our informal notion of an algorithm by a formal definition. Still, the First Incompleteness Theorem is in some sense a computability-theoretic result. Gödel's proof of

the theorem describes an algorithm for constructing a true mathematical statement that is not derivable in a given formal system. Computability theory is very well suited for stating and proving Gödel's Theorem as it is an excellent tool for explicating the computational content of the theorem.

The next lemma bridges the gap between computability theory, on the one hand, and first-order logic, structures, deductions, and formal number theory on the other hand.

Lemma 7.7.1. *For any semi-computable set A, there exists a Σ-formula $\theta(x)$ such that $\mathfrak{N} \models \theta(\overline{a})$ iff $a \in A$ (thus there exists a Σ-formula $\phi(x)$ such that $\mathfrak{N} \models \phi(\overline{a})$ iff $a \in \mathcal{K}$ and, moreover, there exists a Π-formula $\psi(x)$, logically equivalent to $\neg\phi(x)$, such that $\mathfrak{N} \models \psi(\overline{a})$ iff $a \in \overline{\mathcal{K}}$).*

Proof. We will need the following claim:

> **(Claim)** For any computable function $f(\underset{\sim}{x})$, there exists a Σ-formula $\phi(\underset{\sim}{x}, y)$ such that
>
> $$f(\underset{\sim}{a}) = b \quad \Leftrightarrow \quad \mathfrak{N} \models \phi(\underset{\sim}{\overline{a}}, \overline{b}) \; .$$

Before we turn to the proof of the claim, let us ensure that our lemma follows. Let A be a semi-computable set. Thus, we have a computable function f such that $\mathrm{dom}(f) = A$. Let $g(x) = 0 \dot{-} f(x)$. (If $f(a)$ is defined, then $g(a) = 0$. If $f(a)$ is undefined, then $g(a)$ is also undefined.) Now, g is a computable function. Moreover, $g(a) = 0$ iff $a \in A$. By the claim, there exists a Σ-formula $\theta_0(x, y)$ such that $\mathfrak{N} \models \theta_0(\overline{a}, \overline{0})$ iff $a \in A$. Thus, let $\theta(x) :\equiv \theta_0(x, \overline{0})$, and our lemma holds. (Thus, since $\mathcal{K}$ is semi-computable, there will be Σ-formula $\phi(x)$ such that $\mathfrak{N} \models \phi(\overline{a})$ iff $a \in \mathcal{K}$. It is easy to find a Π-formula logically equivalent to $\neg\phi(x)$. Just take the formula $\neg\phi(x)$ and move the negation signs towards the atomic formulas. Use the well-known logical equivalences that are known as DeMorgan's laws: $\neg(\alpha \wedge \beta)$ is equivalent to $(\neg\alpha \vee \neg\beta)$, and $\neg(\exists x)[\alpha]$ is equivalent to $(\forall x)[\neg\alpha]$, and so on. Finally, you cancel double negation signs in front of atomic formulas. The result will be a Π-formula that is logically equivalent to $\neg\phi(x)$.) This shows that the lemma follows from the claim, so we simply have to establish the claim.

We prove the claim by induction on the structure of the computable function f. There will be one case for each clause of Definition 7.2.1.

Case (1): f is the function $\mathcal{S}$. Let $\phi(x, y) :\equiv S(x) = y$, and the claim holds.

Case (2): f is the function $\mathcal{I}_i^n$. Let

$$\phi(x_1, \ldots, x_n, y) \;\; :\equiv \;\; x_1 = x_1 \;\wedge\; \ldots \;\wedge\; x_n = x_n \;\wedge\; x_i = y$$

and the claim holds.

Case (3): f is the function $\mathcal{O}$. Let $\phi(y) :\equiv y = 0$, and the claim holds.

Case (4): $f(\underset{\sim}{x}) = h(g_1(\underset{\sim}{x}), \ldots, g_1(\underset{\sim}{x}))$. By our induction hypothesis we have Σ-formulas $\psi_1, \ldots, \psi_m$ and ξ such that

$$g_i(\underset{\sim}{a}) = b \quad \Leftrightarrow \quad \mathfrak{N} \models \psi_i(\underset{\sim}{\overline{a}}, \overline{b}) \qquad (\text{ for } i = 1, \ldots, m)$$

and

$$h(a_1, \ldots, a_m) = b \quad \Leftrightarrow \quad \mathfrak{N} \models \xi(\overline{a}_1, \ldots, \overline{a}_m, \overline{b}) \; .$$

Let

$$\phi(\underset{\sim}{x}, y) \;\; :\equiv \;\; (\exists z_1) \ldots (\exists z_m)[\psi_1(\underset{\sim}{x}, z_1) \wedge \ldots \wedge \psi_i(\underset{\sim}{x}, z_m) \wedge \xi(z_1, \ldots, z_m, y)] \; .$$

It is easy to see that $\mathfrak{N} \models \phi(\underset{\sim}{\overline{a}}, \overline{b})$ iff $f(\underset{\sim}{a}) = b$. Moreover, ϕ is a Σ-formula as the class of Σ-formulas is closed under conjunctions and existential quantification.

In the next case of our inductive proof, we need the $\mathcal{L}_{NT}$-formula $IthElement(x, y, z)$ from Section 5.6. Recall that $IthElement(\overline{a}, \overline{i}, \overline{t})$ is a Σ-formula which is true (in $\mathfrak{N}$) if and only if a is the number at position i of the sequence encoded by t.

Case (5): $f(\underset{\sim}{x}, 0) = g(\underset{\sim}{x})$ and $f(\underset{\sim}{x}, z{+}1) = h(\underset{\sim}{x}, z, f(\underset{\sim}{x}, z))$. The induction hypothesis yields Σ-formulas $\psi(\underset{\sim}{x}, y)$ and $\xi(\underset{\sim}{x}, z_1, z_2, y)$ such that

$$g(\underset{\sim}{a}) = b \quad \Leftrightarrow \quad \mathfrak{N} \models \psi(\underset{\sim}{\overline{a}}, \overline{b})$$

and

$$h(\underset{\sim}{a}, c_1, c_2) = b \quad \Leftrightarrow \quad \mathfrak{N} \models \xi(\underset{\sim}{\overline{a}}, \overline{c}_1, \overline{c}_2, \overline{b}) \; .$$

Let $\phi(\underset{\sim}{x}, z, y)$ be the formula

$$\begin{aligned}(\exists t)(\exists u_0) \; \big[\; & IthElement(u_0, S(0), t) \; \wedge \\ & \psi(\underset{\sim}{x}, u_0) \;\; \wedge \;\; IthElement(y, S(z), t) \; \wedge \\ & (\forall i < z)(\exists u)(\exists v)[IthElement(u, S(i), t) \wedge \\ & \qquad IthElement(v, S(S(i)), t) \; \wedge \; \xi(\underset{\sim}{x}, i, u, v)] \; \big] \; .\end{aligned}$$

Then, $\phi(\underset{\sim}{\overline{a}}, \overline{c}, \overline{b})$ states that there exists a t that encodes the sequence

$$\Big(\; g(\underset{\sim}{a}) \; , \; \overbrace{h(\underset{\sim}{a}, 0, g(\underset{\sim}{a}))}^{f(\underset{\sim}{a},1)} \; , \; \overbrace{h(\underset{\sim}{a}, 1, h(\underset{\sim}{a}, 0, g(\underset{\sim}{a})))}^{f(\underset{\sim}{a},2)} \; , \; \ldots$$

$$\ldots \; , \; \overbrace{h(\underset{\sim}{a}, c-1, \ldots h(\underset{\sim}{a}, 1, h(\underset{\sim}{a}, 0, g(\underset{\sim}{a}))) \ldots)}^{f(\underset{\sim}{a},c)} \; , \; \ldots \; \Big)$$

where the $(c+1)^{\text{st}}$ element of the sequence is b. Thus, $f(\underset{\sim}{a}, c) = b$ iff $\mathfrak{N} \models \phi(\underset{\sim}{\overline{a}}, \overline{c}, \overline{b})$. Moreover, ϕ is a Σ-formula as the class of Σ-formulas is closed

under existential quantification, bounded quantification, and conjunction. This proves that the claim holds when f is defined by primitive recursion.

Case (6): $f(\underset{\sim}{x}) = (\mu i)[g(\underset{\sim}{x}, i)]$. By the induction hypothesis, we have a Σ-formula $\psi(\underset{\sim}{x}, i, y)$ such that

$$g(\underset{\sim}{a}, i) = b \quad \Leftrightarrow \quad \mathfrak{N} \models \psi(\overline{\underset{\sim}{a}}, \overline{i}, \overline{b}) \; .$$

Let $\phi(\underset{\sim}{x}, y)$ be the formula

$$(\forall i < y)(\exists u)[\psi(\underset{\sim}{x}, i, u) \; \wedge \; \neg(u = 0)] \; \wedge \; \psi(\underset{\sim}{x}, y, 0) \; .$$

The negated atomic formula $\neg(u = 0)$ is a Σ-formula. Furthermore, the class of Σ-formulas is closed under conjunction, existential quantification, and bounded quantification. Thus, we conclude that ϕ is a Σ-formula. It is easy to see that $\mathfrak{N} \models \phi(\overline{\underset{\sim}{a}}, \overline{b})$ iff $f(\underset{\sim}{a}) = b$. This completes the proof of the claim, and hence the lemma. □

We are now at the point where we can settle the Entscheidungsproblem of Hilbert. Recall that, if ϕ is an $\mathcal{L}_{NT}$-formula, then $\ulcorner \phi \urcorner$ is the Gödel number of ϕ, as defined in Section 5.7.

Theorem 7.7.2 (Undecidability of the Entscheidungsproblem). *The set*

$$\{\ulcorner \psi \urcorner \mid \psi \textit{ is a valid } \mathcal{L}_{NT}\textit{-formula}\}$$

is not computable.

Proof. Let $\bigwedge(N)$ denote the conjunction of the axioms of N. First, we prove that there exists an $\mathcal{L}_{NT}$-formula $\phi(x)$ such that

$$a \in \mathcal{K} \quad \Leftrightarrow \quad \models \bigwedge(N) \rightarrow \phi(\overline{a}) \; . \tag{*}$$

Let $\phi(x)$ be the Σ-formula given by Lemma 7.7.1. Then, we have

$$\begin{array}{lll} a \in \mathcal{K} & \Leftrightarrow \quad \mathfrak{N} \models \phi(\overline{a}) & \text{(Lemma 7.7.1)} \\ & \Leftrightarrow \quad N \vdash \phi(\overline{a}) & (N \text{ is } \Sigma\text{-complete. Proposition 5.3.13}) \\ & \Leftrightarrow \quad \vdash \bigwedge(N) \rightarrow \phi(\overline{a}) & \text{(The Deduction Theorem)} \\ & \Leftrightarrow \quad \models \bigwedge(N) \rightarrow \phi(\overline{a}) & \text{(The Completeness Theorem)} \end{array}$$

This proves (*).

Now, assume for the sake of contradiction that the set

$$\{\ulcorner \psi \urcorner \mid \psi \text{ is a valid } \mathcal{L}_{NT}\text{-formula}\}$$

is computable. Then, the set has a computable characteristic function χ. The function $g(x) = \ulcorner \bigwedge(N) \rightarrow \phi(\overline{x}) \urcorner$ is primitive recursive by Lemma 7.3.17. Let $f(x) = \chi(g(x))$. Then, f is a total and computable function. By (*), f is the characteristic function of the set $\mathcal{K}$. But then $\mathcal{K}$ is a computable set. This contradicts Theorem 7.6.12. □

7.7.2 Gödel's First Incompleteness Theorem

Gödel's First Incompleteness Theorem was published in a paper titled "Über formal unentscheidbare Sätze der *Principia Mathematica* und verwandter Systeme." The standard translation into English is "On formally undecidable propositions of *Principia Mathematica* and related systems." *Principia Mathematica* is a thick tome written in the spirit of Hilbert's program. It was written by Alfred North Whitehead and Bertrand Russell and filled three volumes – almost 2000 pages. The title of Gödel's paper indicates that he has proved that the formal calculus of *Principia Mathematica* is incomplete, moreover, and more importantly, the title indicates that his proof will go through for all formal systems that are similar to the one found in *Principia Mathematica*. The incompleteness is not caused by an accidental fault, e.g., it was not the case that Whitehead and Russell had forgotten to add an axiom or two. Gödel proved that any consistent formal system that aims to capture a reasonable portion of mathematics will be incomplete. But he could not really provide an informative and natural definition of a formal system! To complete his proofs, he had to specify certain criteria that a system must satisfy to be counted as a formal system. These criteria became very technical and made the essence of his marvelous results hard to grasp. (Although those of you who have read through Chapters 5 and 6 have seen that truly excellent presentation can help make things easier.)

If we look at the statement of the Incompleteness Theorem in Chapter 6 we can see some of the difficulties:

> Suppose that A is a consistent and recursive set of axioms in the language $\mathcal{L}_{NT}$. Then there is a sentence θ such that $\mathfrak{N} \models \theta$ but $A \nvdash \theta$.

What it means for a set of axioms to be recursive is not easily made formal. The definition of a recursive set is very technical, and thus, it requires a considerable amount of effort to realize why any system we intuitively conceive as a formal system indeed will be incomplete. In the setting of computability theory, we can formulate the insights buried in Gödel's work on incompleteness more naturally: There is no need to talk about recursive sets of formulas anymore. Instead we can talk about computable sets of formulas – and semi-computable sets of formulas. These are intuitively clear notions. A set of formulas is computable when there is an algorithm for deciding if a formula is in the set. A set of formulas is semi-computable when there is an algorithm for confirming that a formula is in the set.

We will prove several versions of Gödels First Incompleteness Theorem. Let us start off with an informal outline of a proof of a weak version of the theorem: Lemma 7.7.1 yields an $\mathcal{L}_{NT}$-formula $\psi(x)$ such that $\mathfrak{N} \models \psi(\overline{a})$ iff $a \in \overline{\mathcal{K}}$. Now, assume that we have a set A of $\mathcal{L}_{NT}$-axioms such that the set $\{\ulcorner \eta \urcorner \mid A \vdash \eta\}$ is semi-computable. Furthermore, assume that $\mathfrak{N} \models A$. If

$A \vdash \psi(\overline{17})$, what do we know? Well, by the Soundness Theorem, we know that $\mathfrak{N} \models \psi(\overline{17})$ and, hence, we also know that $17 \in \overline{\mathcal{K}}$. Thus, if we can deduce $\psi(\overline{17})$ from our axioms, we know that 17 is in the set $\overline{\mathcal{K}}$. Similarly, if we can deduce $\psi(\overline{31425})$ from the axioms, we know that 31425 is in $\overline{\mathcal{K}}$. But we cannot possibly derive $\psi(\overline{a})$ for every $a \in \overline{\mathcal{K}}$. If we could do that, the set $\overline{\mathcal{K}}$ would be semi-computable, and this would contradict Theorem 7.6.11. Thus, there exists a least one $a \in \mathbb{N}$ such that $\mathfrak{N} \models \psi(\overline{a})$ and $A \nvdash \psi(\overline{a})$.

Before we can prove our weak version of Gödel's theorem, we need a straightforward technical lemma.

Lemma 7.7.3. *Let $\phi(x)$ be an $\mathcal{L}_{NT}$-formula, and let A be a set of $\mathcal{L}_{NT}$-axioms such that $\{\ulcorner \eta \urcorner \mid A \vdash \eta\}$ is a semi-computable set. Then, $\{a \mid A \vdash \phi(\overline{a})\}$ is also a semi-computable set.*

Proof. Since $\{\ulcorner \eta \urcorner \mid A \vdash \eta\}$ is semi-computable, we have a computable function f such that $\mathrm{dom}(f) = \{\ulcorner \eta \urcorner \mid A \vdash \eta\}$. Let $g(a) = \ulcorner \phi(\overline{a}) \urcorner$ and let $h(x) = f(g(x))$. The function g is primitive recursive by Lemma 7.3.17 and, thus, h is a computable function. We conclude that $\{a \mid A \vdash \phi(\overline{a})\}$ is a semi-computable set since $\mathrm{dom}(h) = \{a \mid A \vdash \phi(\overline{a})\}$. □

Theorem 7.7.4 (Incompleteness Version I). *Let A be a set of $\mathcal{L}_{NT}$-axioms such that $\mathfrak{N} \models A$ and the set $\{\ulcorner \eta \urcorner \mid A \vdash \eta\}$ is semi-computable. Then, there is a Π-sentence θ such that $\mathfrak{N} \models \theta$ but $A \nvdash \theta$.*

Proof. By Lemma 7.7.1, we have a Π-formula $\psi(x)$ such that $a \in \overline{\mathcal{K}}$ iff $\mathfrak{N} \models \psi(\overline{a})$. Now, $\mathfrak{N}$ is a model for A, and by the Soundness Theorem, we have

$$\{a \mid A \vdash \psi(\overline{a})\} \subseteq \{a \mid \mathfrak{N} \models \psi(\overline{a})\} = \overline{\mathcal{K}}\,. \qquad (*)$$

The set $\{a \mid A \vdash \psi(\overline{a})\}$ is semi-computable by Lemma 7.7.3, and the set $\overline{\mathcal{K}}$ is not semi-computable by Theorem 7.6.11. Thus, $\overline{\mathcal{K}}$ cannot equal the set $\{a \mid A \vdash \psi(\overline{a})\}$, and the inclusion in (*) has to be strict. For at least one $a \in \mathbb{N}$, we have $\mathfrak{N} \models \psi(\overline{a})$ and $A \nvdash \psi(\overline{a})$. Thus, let $\theta :\equiv \psi(\overline{a})$, and the theorem holds. □

Our next version of Gödel's theorem is stronger than the one above. In the version above we assume $\mathfrak{N}$ is a model for the axioms. In the version below we will only assume that the axioms are consistent. This is a weaker assumption as the Soundness Theorem says that any set of formulas having a model is consistent. Moreover, it is a strictly weaker assumption as $\mathfrak{N}$ is not necessarily a model for a consistent set of $\mathcal{L}_{NT}$-formulas.

It was the American mathematician John Barkley Rosser who proved that it was sufficient to assume that the axioms were consistent. Gödel himself assumed that the axioms were ω-consistent. That was weaker than assuming that $\mathfrak{N}$ was a model for the axioms, but not as weak as assuming that the axioms were consistent.

Theorem 7.7.5 (Incompleteness Version II). *Let A be a set of $\mathcal{L}_{NT}$-axioms such that A is consistent and the set $\{\ulcorner \eta \urcorner \mid A \vdash \eta\}$ is semi-computable. Then, there is a Π-sentence θ such that $\mathfrak{N} \models \theta$ but $A \nvdash \theta$.*

Proof. We split the proof in two cases: (i) The case when the theory A extends the theory N, that is, every formula derivable from the axioms of N is also derivable from the axioms of A. (ii) The case when the theory A does not extend the theory N, that is, we have $N \vdash \phi$ and $A \nvdash \phi$ for some ϕ.

Case (ii) is easy. If A could prove all the axioms of N, then A would extend N. Hence, at least one the eleven axioms of of N cannot be derived from the axioms in A. All the axioms are Π-sentences. Let θ be the conjunction of the eleven axioms of N. Then, θ is a Π-sentence such that $\mathfrak{N} \models \theta$ and $A \nvdash \theta$.

We turn to case (i). By Lemma 7.7.1, we have a Σ-formula $\phi(x)$ such that $a \in \mathcal{K}$ iff $\mathfrak{N} \models \phi(\overline{a})$. Moreover,

$$\begin{array}{lcll} a \in \mathcal{K} & \Leftrightarrow & \mathfrak{N} \models \phi(\overline{a}) & \text{(Lemma 7.7.1)} \\ & \Leftrightarrow & N \vdash \phi(\overline{a}) & (N \text{ is } \Sigma\text{-complete. Proposition 5.3.13}) \\ & \Rightarrow & A \vdash \phi(\overline{a}) & (A \text{ extends } N) \end{array}$$

We see that A proves $\phi(\overline{a})$ for every $a \in \mathcal{K}$. Now, A is consistent. There is no η such that $A \vdash \eta$ and $A \vdash \neg\eta$. Thus, if A proves $\neg\phi(\overline{a})$, it cannot be the case that a is in $\mathcal{K}$. Hence, if $A \vdash \neg\phi(\overline{a})$, then $a \in \overline{\mathcal{K}}$. Lemma 7.7.1 states that there is a Π-formula $\psi(\overline{a})$ which is logically equivalent $\neg\phi(\overline{a})$. By the Soundness and Completeness Theorems for first-order logic, $A \vdash \psi(\overline{a})$ iff $A \vdash \neg\phi(\overline{a})$. Hence,

$$\{a \mid A \vdash \psi(\overline{a})\} \subseteq \{a \mid \mathfrak{N} \models \psi(\overline{a})\} = \overline{\mathcal{K}}\,. \qquad (*)$$

From (*) we can proceed as we did in the proof of Version I and conclude that there exists a Π-sentence θ such that $\mathfrak{N} \models \theta$ and $A \nvdash \theta$. □

In Theorem 7.7.4, and in Theorem 7.7.5, we require the set of (Gödel numbers of) formulas derivable from the axioms to be semi-computable. In our version of the First Incompleteness Theorem in Chapter 6, we required the set of axioms to be recursive. That is equivalent to requiring the set of axioms to be computable. (It can be proven that a set is computable if, and only, if it is recursive.) The reader should note that the set of derivable formulas will be semi-computable whenever the set of axioms is semi-computable. Thus, if the set of axioms is computable, then the set of derivable formulas will be semi-computable. And even if the set of axioms is not computable, the set of derivable formulas may still be semi-computable since there exists semi-computable sets that are not computable. The set of

all Σ-sentences true in the standard model $\mathfrak{N}$, is a natural example of a non-computable set of axioms that is semi-computable (see Exercise 2). This means that Theorem 7.7.5 is slightly stronger than our original statement of Gödel's result in Theorem 6.3.7.

The Π-sentence θ that occurs in the versions of Gödel's theorem given above, is often called a *Gödel sentence*. When the set of axioms A satisfies certain conditions, we have $\mathfrak{N} \models \theta$ and $A \nvdash \theta$. Theorem 7.7.5 asserts that there is a Gödel sentence θ, but does not say anything about how we can find this θ. If we examine the proof, we will get more information. We can see that the Gödel sentence will be of the form $\psi(\overline{a})$, where $\psi(\overline{a})$ is true (in $\mathfrak{N}$) if and only if the natural number a is in the set $\overline{K}$. However, the proof does not yield a particular a. The argument in the proof allows us to conclude that such an a exists, but the argument does not tell us how to find this a.

The proof of our next, and last, version of Gödel's First Incompleteness Theorem tells us how we can find a Gödel sentence for a given set of axioms. The theorem states the existence of primitive recursive function $\imath$ (and the proof tells us how to define $\imath$). We can find (the Gödel number of) a Π-sentence θ such that $\mathfrak{N} \models \theta$ and $A \nvdash \theta$ by computing the value of $\imath(e)$ for an e such that $W_e = \{\ulcorner \eta \urcorner \mid A \vdash \eta\}$. Observe that such an index e is nothing but an encoded description of an algorithm for confirming (semi-deciding) if there exists an A-deduction of a formula η (we have $\ulcorner \eta \urcorner \in \mathrm{dom}(\{e\})$ iff $A \vdash \eta$). Thus, a slightly informal, but very reasonable, interpretation of our strongest version of Gödel's theorem is that by following an algorithm we can convert

> an algorithm that halts on input η iff there exists an A-deduction of η

into a Π-sentence θ such that

> if A is consistent, then $\mathfrak{N} \models \theta$ and $A \nvdash \theta$.

To prove the theorem, we need a stronger version of Lemma 7.7.3.

Lemma 7.7.6. *Let $\phi(x)$ be an $\mathcal{L}_{NT}$-formula, and let A be a set of $\mathcal{L}_{NT}$-axioms. There is a primitive recursive function $\jmath$ with the following property: If $W_e = \{\ulcorner \eta \urcorner \mid A \vdash \eta\}$, then $W_{\jmath(e)} = \{a \mid A \vdash \phi(\overline{a})\}$.*

Proof. Let $W_e = \{\ulcorner \eta \urcorner \mid A \vdash \eta\}$. Then, $\mathrm{dom}(\{e\}) = \{\ulcorner \eta \urcorner \mid A \vdash \eta\}$. Let $g(a) = \ulcorner \phi(\overline{a}) \urcorner$ and let $h(x) = \{e\}(g(x))$. The function g is primitive recursive by Lemma 7.3.17, and thus, h is a computable function. We can conclude that $\{a \mid A \vdash \phi(\overline{a})\}$ is a semi-computable set since $\mathrm{dom}(h) = \{a \mid A \vdash \phi(\overline{a})\}$.

So far, we have more or less just repeated the proof of Lemma 7.7.3. Next, we need to prove that an index for h can be computed primitive recursively from e, that is, we need to prove that there is primitive recursive

function $\jmath$ such that $\{\jmath(e)\}(x) = h(x)$ (for all $x \in \mathbb{N}$). Then our proof will be complete as $W_{\jmath(e)} = \text{dom}(\{\jmath(e)\}) = \text{dom}(h) = \{a \mid A \vdash \phi(\overline{a})\}$.

We define the function h_0 by

$$h_0(y, x) \;=\; \mathcal{U}((\mu t)[\mathcal{T}(y, g(x), t)]) \;=\; \{y\}(g(x)) \qquad (*)$$

The last equality holds by Definition 7.4.4. Obviously, h_0 is a computable function, and thus, h_0 has a computable index. Fix such an index d for h_0, and let $\jmath(y) = S_1^1(d, y)$ where S_1^1 is given by the S-m-n Theorem (Theorem 7.4.9). Then, $\jmath$ is a primitive recursive function. Moreover, we have

$$\begin{array}{lcll} \{\jmath(y)\}(x) & = & \{S_1^1(d, y)\}(x) & \text{def. of } \jmath \\ & = & \{d\}(y, x) & \text{the S-m-n Theorem} \\ & = & h_0(y, x) & d \text{ is an index for } h_0 \\ & = & \{y\}(g(x))\,. & (*) \end{array}$$

Hence, $\{\jmath(e)\}(x) = \{e\}(g(x)) = h(x)$. □

Theorem 7.7.7 (Incompleteness Version III). *There exists a primitive recursive function $\imath$ with the following property: If A is a consistent set of $\mathcal{L}_{NT}$-axioms and e is an index such that $\mathcal{W}_e = \{\ulcorner \eta \urcorner \mid A \vdash \eta\}$, then $\imath(e)$ is the Gödel number of a Π-sentence θ such that $\mathfrak{N} \models \theta$ and $A \nvdash \theta$.*

Proof. First, we assume that A is a consistent set of $\mathcal{L}_{NT}$-axioms and that e is an index such that $\mathcal{W}_e = \{\ulcorner \eta \urcorner \mid A \vdash \eta\}$.

Let $\bigwedge(N)$ be the conjunction of the eleven axioms of the theory N. Then, $\bigwedge(N)$ is a Π-sentence such that $\mathfrak{N} \models \bigwedge(N)$, and as we saw in the proof of Version II: If A does not extend N, then $A \nvdash \bigwedge(N)$.

Now, assume that A extends N. Since A is a consistent set of axioms, we can proceed, as we did in the proof of Version II, and establish the inclusion

$$\{a \mid A \vdash \psi(\overline{a})\} \;\subseteq\; \{a \mid \mathfrak{N} \models \psi(\overline{a})\} \;=\; \overline{\mathcal{K}} \qquad (*)$$

where $\psi(x)$ is a Π-sentence. Now $\mathcal{W}_e = \{\ulcorner \eta \urcorner \mid A \vdash \eta\}$. By (*) and Lemma 7.7.6, we have a primitive recursive function $\jmath$ such that

$$W_{\jmath(e)} \;=\; \{a \mid A \vdash \psi(\overline{a})\} \;\subseteq\; \overline{\mathcal{K}}\,. \qquad (**)$$

Let $b = \jmath(e)$. (The natural number b exists as $\jmath$ is a primitive recursive, and thus a total computable, function.) By (**) and Lemma 7.6.13, we have $b \in \overline{\mathcal{K}} \setminus W_b$. Thus, we have $\mathfrak{N} \models \psi(\overline{b})$ as $b \in \overline{\mathcal{K}}$, and we have $A \nvdash \psi(\overline{b})$ as $b \notin W_b = \{a \mid A \vdash \psi(\overline{a})\}$.

Our theorem asserts the existence of a primitive recursive function $\imath$. Let

$$\imath(e) \;=\; \ulcorner \bigwedge(N) \;\wedge\; \psi(\overline{\jmath(e)}) \urcorner$$

and the theorem holds. The function $\imath$ is primitive recursive by Lemma 7.3.17, and $\bigwedge(N) \wedge \psi(\overline{\jmath(e)})$ is a Π-sentence (the conjunction of two Π-sentences is a Π-sentence). Furthermore, given that A is a consistent set of axioms and that $\mathcal{W}_e = \{\ulcorner \eta \urcorner \mid A \vdash \eta\}$, we have

$$\mathfrak{N} \models \bigwedge(N) \wedge \psi(\overline{\jmath(e)}) \qquad \text{and} \qquad A \not\vdash \bigwedge(N) \wedge \psi(\overline{\jmath(e)}) \,.$$

(If A does not extend N, then $A \not\vdash \bigwedge(N)$. If A extends N, then $A \not\vdash \psi(\overline{\jmath(e)})$.) □

From an intuitive computability-theoretic point of view, Gödel's First Incompleteness Theorem is an inevitable consequence of the fact that we can define an undecidable set in the $\mathcal{L}_{NT}$-structure $\mathfrak{N}$. (There is a formula $\phi(x)$ such that $\mathfrak{N} \models \phi(\overline{a})$ iff $a \in \mathcal{K}$.) Since we can define an undecidable set in $\mathfrak{N}$, no semi-decidable set of $\mathcal{L}_{NT}$-axioms will be complete for $\mathfrak{N}$. If there were such a set of axioms, we could decide membership in an undecidable set. (We could decide if a is a member of $\mathcal{K}$ by enumerating deductions until we encountered a deduction of $\phi(\overline{a})$ or a deduction of $\neg\phi(\overline{a})$.)

The expressive power (the standard interpretation) of the language $\mathcal{L}_{NT}$ is essential. To define an undecidable set like $\mathcal{K}$, we need an expressive language. In the next section we will explain why we still will be able to define $\mathcal{K}$ in the standard structure even if the language just contains $0, S, +, \cdot$ (zero, successor, addition, multiplication). Thus, the Gödel's First Incompleteness Theorem holds when $\mathfrak{N}$ is the standard structure for the language $\{0, S, +, \cdot\}$.

First-order number theory over the language $\{0, S, +, <\}$ (and a standard structure $\mathfrak{N}$) is called *Presburger Arithmetic*. There is no symbol for multiplication – and no symbol for exponentiation – in the language and, thus, Presburger Arithmetic has limited expressive power. For example, we cannot define the set of primes – or the set of all powers of 2 – in Presburger Arithmetic. Neither can we define the set $\mathcal{K}$ or any other undecidable set. There exists a semi-decidable set of axioms A such that for any formula ϕ of *Presburger Arithmetic* we have $\mathfrak{N} \models \phi$ iff $A \vdash \phi$. Gödel's First Incompleteness Theorem does not hold for Presburger Arithmetic.

7.7.3 Exercises

1. Let A be a consistent set of $\mathcal{L}_{NT}$-axioms, and let

   - $U = \{\ulcorner \phi \urcorner \mid \phi \in A\}$
   - $V = \{\ulcorner \phi \urcorner \mid A \vdash \phi\}$.

   (a) Discuss the five assertions below. Which ones are true, and which ones are false? Be careful! You may encounter some tricky questions.

   (1) If U is a computable set, then V is a computable set.

(2) If U is a computable set, then V is a semi-computable set.
(3) If U is a semi-computable set, then V is a semi-computable set.
(4) If V is a computable set, then U is a computable set.
(5) If V is a semi-computable set, then U is a semi-computable set.

(b) Explain why there exists a primitive recursive function $\imath$ with the following property:

$$\mathcal{W}_e = U \quad \Rightarrow \quad \mathcal{W}_{\imath(e)} = V \ .$$

2. Let $A = \{\ulcorner\phi\urcorner \mid \phi \text{ is a } \Sigma\text{-sentence and } \mathfrak{N} \models \phi\}$. Prove that A is a semi-computable set. Prove that A is not a computable set.

3. Let $A = \{\phi \mid \phi \text{ is a } \Pi\text{-sentence and } \mathfrak{N} \models \phi\}$. Why does it not follow from Gödel's First Incompleteness Theorem that there exists an $\mathcal{L}_{NT}$-sentence θ such that $\mathfrak{N} \models \theta$ and $A \nvdash \theta$?

4. Let $A = \{\phi \mid \phi \text{ is a } \Pi\text{-sentence and } \mathfrak{N} \models \phi\}$. (Note that all the axioms of N are elements of the set A.) Prove that there exists an $\mathcal{L}_{NT}$-sentence θ such that $\mathfrak{N} \models \theta$ and $A \nvdash \theta$. This θ can obviously not be a Π-sentence. Can you say something about the complexity of θ in terms of quantifiers occurring in θ? (This is a very hard exercise.)

5. We know that we can write up a deduction of any valid first-order formula ϕ. But how lengthy will these deductions be? Well, they can be *really* long. The shortest possible deduction of ϕ may be so long that it never will be found – neither by man nor by machine.

   Let $\ell(\phi)$ denote the number of symbols in the shortest possible deduction of ϕ (we have no non-logical axioms). Let f be a total computable function (so f may for example be the Ackermann function). Prove that there exists an $\mathcal{L}_{NT}$-formula θ such that $f(\ulcorner\theta\urcorner) < \ell(\theta)$.

7.8 More on Undecidability

In the previous section we proved that the Entscheidungsproblem is undecidable. In this section we will study a couple of other classical undecidability results.

One of the most celebrated results of computability theory is the negative solution of Hilbert's 10th Problem. This is an undecidability result that involves one of the core areas of classical mathematics: polynomial equation in two or more unknowns. Such equations are named after the the Hellenistic mathematician Diophantus of Alexandria who lived in the third century CE. They are called *Diophantine equations*. Consider the equation

$$2y(x+1)^2 \ - \ 4z^3 \ = \ 4xy \ + \ 2y \ - \ 7 \ .$$

This equation can be written in the form

$$2x^2y^1z^0 \ + \ (-4)x^0y^0z^3 \ + \ 7x^0y^0z^0 \ = \ 0 \, .$$

The equation $x^2 + y^2 = 2$ can be written in the form

$$x^2y^0 \ + \ x^0y^2 \ + \ (-2)x^0y^0 \ = \ 0.$$

A Diophantine equation is an equation that can be written in the form

$$c_1 x_1^{a_1^1} \cdots x_n^{a_n^1} \ + \ c_2 x_1^{a_1^2} \cdots x_n^{a_n^2} \ + \ \ldots \ + \ c_m x_1^{a_1^m} \cdots x_n^{a_n^m} \ = \ 0$$

where $x_1, \ldots, x_n$ are unknowns, $c_1, \ldots, c_m$ are coefficients and $a_i^j \in \mathbb{N}$ (for $i = 1, \ldots, n$ and $j = 1, \ldots, m$). You should think of the coefficients as given and the unknowns as the quantities we want to determine. When we have found values for the unknowns which satisfy the equation, we have a solution of the equation. The Diophantine equation $x^2 + y^2 = 1$ has two solutions in $\mathbb{N}^2$: $x = 0$ and $y = 1$, and $x = 1$ and $y = 0$. There are no other solutions of the equation in the natural numbers. If we allow for solutions in the integers, we can find two more solutions: $x = 0$ and $y = -1$, and $x = -1$ and $y = 0$. (The equation has of course infinitely many solutions in the real numbers.) The Diophantine equation $x^3 - y^2 = 0$ has infinitely many solutions in the natural numbers. For each $i \in \mathbb{N}$, we have a solution when $x = i^2$ and $y = i^3$. The Diophantine equation $x+y+z = x+y+z+1$ has no solutions.

In 1900, at the International Congress of Mathematicians in Paris, Hilbert outlined 23 problems which he predicted would shape the next century of mathematics. The 10th problem on Hilbert's list can be stated as follows:

> Derive an algorithm that decides if a Diophantine equation with integer coefficients (the equation is the input to the algorithm) has a solution in the natural numbers.

Well, Hilbert did not use the word "algorithm." He asked for "a process according to which it can be determined in a finite number of operations..." But what he meant was what we today call an algorithm. Neither did he ask for an algorithm that decides if the equation has a solution in the natural numbers. He asked for one that decides if the equation has a solution in the integers. But if you have an algorithm that can decide if there is a solution in the naturals, then you can easily construct an algorithm that decides if there is a solution in the integers; and vice versa, if you have an algorithm that can decide if there is a solution in the integers, then you can easily construct an algorithm that decides if there is a solution in the naturals. So it is all right to state Hilbert's 10th problem the way we have done above.

It is a consequence of the next theorem that the algorithm that Hilbert asked for in his 10th problem does not exist.

Theorem 7.8.1 (Matiyasevich-Robinson-Davis-Putnam, MRDP). *For any semi-computable set A there exists a Diophantine equation*

$$p(y, x_1, \ldots, x_n) = 0$$

with integer coefficients such that

$$a \in A \;\Leftrightarrow\; p(a, x_1, \ldots, x_n) = 0 \text{ has a solution in the natural numbers.}$$

It is easy to see that the unsolvability of the 10th problem follows from this theorem: Assume that the algorithm Hilbert asked for exists. By the theorem we have an equation $p_{\mathcal{K}}(y, x_1, \ldots, x_n) = 0$ such that $a \in \mathcal{K}$ if and only if $p_{\mathcal{K}}(a, x_1, \ldots, x_n) = 0$ has a solution in the natural numbers. Now, $p_{\mathcal{K}}(a, x_1, \ldots, x_n) = 0$ is a Diophantine equation with integer coefficients since $p_{\mathcal{K}}(y, x_1, \ldots, x_n) = 0$ is a Diophantine equation with integer coefficients. So we can ask the algorithm if $p_{\mathcal{K}}(a, x_1, \ldots, x_n) = 0$ has a solution in the natural numbers. If the algorithm says YES, we know that $a \in \mathcal{K}$. If the algorithm says NO, we know that $a \notin \mathcal{K}$. Hence, $\mathcal{K}$ is a computable set. But we know (Theorem 7.6.12) that $\mathcal{K}$ is not a computable set.

Let $\mathcal{L}^-_{NT}$ be the first-order language $\{0, S, +, \cdot\}$. Then, $\mathcal{L}^-_{NT}$ is a subset of the language of number theory $\mathcal{L}_{NT}$. It is not hard to see that every Diophantine equation with integer coefficients can be expressed by an atomic $\mathcal{L}^-_{NT}$-formula (in the standard model $\mathfrak{N}$). E.g., the Diophantine equation $3x^2 - y^3 - 2 = 0$ can be expressed by the formula

$$(x \cdot x) + (x \cdot x) + (x \cdot x) \;\; = (y \cdot y \cdot y) + SS0\,.$$

Moreover, an atomic $\mathcal{L}^-_{NT}$-formula is of the form $t_1 = t_2$ where t_1 and t_2 are $\mathcal{L}^-_{NT}$-terms. Hence, an atomic $\mathcal{L}^-_{NT}$-formula is nothing but a Diophantine equation when it is interpreted in the standard model $\mathfrak{N}$. This means that Theorem 7.8.1, henceforth referred to as the MRDP Theorem, is trivially equivalent to the following theorem.

Theorem 7.8.2. *For any semi-computable set A there exists an atomic $\mathcal{L}^-_{NT}$-formula $\phi(y, \underset{\sim}{x})$ such that $a \in A$ iff $\mathfrak{N} \models (\exists \underset{\sim}{x})\phi(\overline{a}, \underset{\sim}{x})$.*

Observe the similarity between Theorem 7.8.2 and our Lemma 7.7.1. This is the lemma stating that for any semi-computable set A there is an Σ-formula $\psi(y)$ such that $a \in A$ iff $\mathfrak{N} \models \psi(\overline{a})$. Now, the formula $(\exists \underset{\sim}{x})\phi(\overline{a}, \underset{\sim}{x})$ in Theorem 7.8.2 is a Σ-formula and, thus, Lemma 7.7.1 follows trivially from the MRDP Theorem. Moreover, when Lemma 7.7.1 is given, the MRDP Theorem becomes equivalent to the next theorem.

Theorem 7.8.3. *For any Σ-formula ψ there exists an atomic $\mathcal{L}^-_{NT}$-formula ϕ such that $\mathfrak{N} \models \psi$ iff $\mathfrak{N} \models (\exists \underset{\sim}{x})\phi$.*

Now it is pretty easy to see what we need to do to prove the MRDP Theorem. A Σ-formula is a number-theoretic formula that may contain

bounded quantifiers, propositional connectives, the ordering relation $<$, and the exponential function E. To prove the MRDP Theorem, we have to prove that any Σ-formula can be expressed by a formula of much simpler form – a formula that does not contain bounded quantifiers, connectives, and so on – a formula that is simply a single polynomial equation preceded by a row of existential quantifiers.

The proof of the MRDP Theorem requires a fair amount of nontrivial number theory, and we will not give the full proof here. The theorem was initially a conjecture posed by Martin Davis in 1949. A few years later, he was able to prove that any Σ-formula is equivalent to a formula of the form

$$(\exists y)(\forall z \leq y)(\exists x_1, \ldots x_n)\phi(y, z, x_1, \ldots x_n)$$

where ϕ is an atomic $\mathcal{L}^-_{NT}$-formula. It looked like Davis was not very far from having a proof of the theorem – only one bounded universal quantifier stood in the way. But it took more than 15 years to get rid of that quantifier. Several important relevant results were proved in the meantime by Yuri Matiyasevich, Julia Robinson, Hilary Putnam, and Davis himself, but the proof was not complete until 1970. It was Matiyasevich who provided the last piece of the puzzle.

The MRDP Theorem is a fantastic result. When Davis conjectured the theorem, his conjecture was considered to be quite bold. Many mathematicians found the consequences implausible. For example, the conjecture implied that there is a Diophantine equation $p(y, x_1, \ldots, x_n) = 0$ such that $p(a, x_1, \ldots, x_n) = 0$ has a solution iff a is a prime; the conjecture implied that there is a Diophantine equation $p(y, x_1, \ldots, x_n) = 0$ such that $p(a, x_1, \ldots, x_n) = 0$ has a solution iff a is a power of 2. No one had ever seen such equations. Moreover, the conjecture implied that there exists a particular Diophantine equation $p_0(u, y_1, \ldots, y_m) = 0$ with integer coefficients that has a seemingly marvelous property: For any Diophantine equation $p(x_1, \ldots, x_n) = 0$ with integer coefficients, there exists $k \in \mathbb{N}$ such that

$$p(x_1, \ldots, x_n) = 0 \text{ has a solution in } \mathbb{N} \quad \Leftrightarrow \quad p_0(k, y_1, \ldots, y_m) = 0 \text{ has a solution in } \mathbb{N}\,.$$

Note that the equation $p(x_1, \ldots, x_n) = 0$ may be of any degree and have any number of unknowns, whereas $p_0(u, y_1, \ldots, y_m) = 0$ is a particular equation of a fixed degree and with a fixed number of unknowns.

> *Chaff:* Think about this one for a minute. This universal polynomial p_0 might be of degree 10^{10} and have 4215 variables, but it can sniff out whether *any* polynomial, of *any* degree, in *any* number of variables has a solution in $\mathbb{N}$. That's just weird.

When Davis made his conjecture, it was hard to believe that here could be Diophantine equations that could pick out such sets. But the conjecture

was turned into a theorem, and these equations do definitely exist (see Exercise 6).

The MRDP Theorem also entails that there exist Gödel sentences of a very simple form. By the theorem, there exists an $\mathcal{L}^{-}_{NT}$-formula $\phi(x)$ of the form

$$\forall y_1, \ldots y_n[\neg t_1 = t_2]$$

such that $a \in \overline{\mathcal{K}}$ iff $\mathfrak{N} \models \phi(\overline{a})$. From this, our theorems and proofs of Section 7.7.1 quickly yield that there is a Gödel sentence ϕ_A of this simple form for any semi-decidable set of $\mathcal{L}_{NT}$-axioms A.

There should be no need to elaborate on why undecidability results are of interest to a wide range of scientists and scholars. Many natural problems in Analysis, Topology, Matrix Theory, Graph Theory, Group Theory, Algebraic Geometry, ... yes, we can go on for quite a while ... Number Theory, Automata Theory, Formal Language Theory, Complexity Theory, Program Verification Theory ... have been proved to be undecidable. We will round off this chapter on computability theory by discussing and proving an undecidability result of a very general nature.

Theorem 7.8.4 (Rice's Theorem). *Let $\mathcal{A}$ be a set of unary computable functions, and let A be the index set of $\mathcal{A}$, that is, $A = \{e \mid \{e\} \in \mathcal{A}\}$. Furthermore, assume that $\mathcal{A}$ is not empty, and assume that $\mathcal{A}$ does not contain all the computable functions. Then, A is not a computable set.*

Probably the best way to understand what Rice's Theorem says is to study a few of its corollaries.

Corollary 1: It is not decidable if a given number e is an index for the identity function (that is, the set $\{e \mid \{e\}(x) = x \text{ for all } x \in \mathbb{N}\}$ is not computable).

Corollary 2: It is not decidable if two given numbers e and d are indices for the same computable function. In other words, the set

$$\{\langle e, d\rangle \mid \{e\}(x) = \{d\}(x) \text{ for all } x \in \mathbb{N}\}$$

is not computable.

Corollary 3: It is not decidable if a given number e is an index for a total function (that is, the set $\{e \mid \text{dom}(\{e\}) = \mathbb{N}\}$ is not computable).

Corollary 4: It is not decidable if a given number e is an index for a function that belongs to the 17th Grzegorczyk class (the class usually denoted $\mathcal{E}^{17}$).

To see that the first corollary follows from the theorem, let $\mathcal{A}$ be the set that contains the identity function, and nothing but the identity function, and let A be the index set of $\mathcal{A}$.

> *Chaff:* Once again, we have to be picky about fonts and typefaces. Be careful about A vs. $\mathcal{A}$.

According to Rice's Theorem, A is not a computable set. Thus, we cannot decide if a given number e is an index for the identity function (simply because A is the set of indices for the identity function). To see that the second corollary follows, pick an arbitrary index d, and let $\mathcal{A}_d$ be the set that contains nothing but the function $\{d\}$. Furthermore, let A_d be the index set of $\mathcal{A}_d$. If you can decide if two given indexes e and d are indices for the same function, then A_d is a computable set. But according to Rice's Theorem, A_d is not a computable set. Hence, the second corollary holds. The third corollary follows straightforwardly: Let A be the index set of the set of all total functions. By the theorem, A is not a computable set. We leave the proof of the fourth corollary to the reader. If you cannot recall the definition of the 17th Grzegorczyk class, or if you have never heard of it, do not worry. It is sufficient to know that this is a class of computable functions, and that some functions will be in the class, and that some functions will not be in the class.

The moral of Rice's Theorem is simply this: When we are given a computable index e, then any question regarding the function $\{e\}$ is undecidable! (Unless we ask a question to which the answer is YES for any $e \in \mathbb{N}$ or a question to which the answer is NO for any $e \in \mathbb{N}$. But we do not want ask such a question, do we?) A computable index can be viewed as program code written in a programming language like Java or Pascal. A computer can of course execute such code, but it cannot answer questions about the functions that will be computed when the code is executed. Does this program compute a total function? Is it possible that this program will output 0 for some input? Does this program terminate if the input satisfies certain criteria? By Rice's Theorem, all such questions are undecidable when the code is written in a general programming language like Java or C or Haskell or Python or Prolog or .... This has implication for software development in the information technology industry: Certain necessary engineering tasks can never be fully automatized.

To prepare ourselves for the proof of Rice's Theorem, we will outline an informal and intuitive proof of one of the corollaries of the theorem. The proof will be direct, in the sense that it is not based on Rice's Theorem.

This is a general recipe for undecidability proofs:

- We want to prove that a problem P is undecidable.
- We take a problem Q that we already know is undecidable.

- We reduce the decidability of Q to the decidability of P, that is, we prove that
$$P \text{ is decidable} \quad \Rightarrow \quad Q \text{ is decidable .}$$
- Now, we can conclude that P is undecidable (if P were decidable, the Q would also be decidable).

We followed this recipe when we proved the Entscheidungsproblem to be undecidable. We reduced the problem of deciding membership in $\mathcal{K}$ (a problem we knew was undecidable) to the Entscheidungsproblem. We will now follow this same recipe to prove that it is not decidable whether a number e is an index for the identity function. More formally,

$$\text{Let } A = \{e \mid \{e\}(x) = x \text{ for all } x \in \mathbb{N}\}. \text{ The set } A \text{ is not a computable set.} \tag{*}$$

In order to prove (*), we will reduce a problem that we know is undecidable to the problem of deciding membership in A. Again we will use the "mother of all undecidable problems," namely the problem of deciding membership in $\mathcal{K}$. We will carry out the reduction by providing a total computable function $\imath$ such that

$$a \in \mathcal{K} \quad \text{if and only if} \quad \imath(a) \in A. \tag{†}$$

Once we have produced this function $\imath$, we will be more or less finished, as it is easy to see that any algorithm for deciding membership in A would yield an algorithm for deciding membership in $\mathcal{K}$: Given $a \in \mathbb{N}$, to decide whether $a \in \mathcal{K}$, we can simply...

- compute the value v of $\imath(a)$
- ask the algorithm that decides membership in A if $v \in A$
- if the answer is YES, then $a \in \mathcal{K}$; if the answer is NO, then $a \notin \mathcal{K}$.

Thus, if we could decide membership in A, then we could decide the undecidable problem of membership in $\mathcal{K}$. Therefore we cannot decide membership in A, and so A is not a computable set.

So, to finish the proof we must find a total computable function $\imath$ that satisfies the condition (†). We know that the set $\mathcal{K}$ is semi-decidable, and thus there is a computable function h such that $\mathrm{dom}(h) = \mathcal{K}$. For each $a \in \mathbb{N}$, define the algorithm $\texttt{ALG}_a$:

- input: x
- compute the number $h(a)$ (this computation terminates iff $a \in \mathcal{K}$)
- give the output x.

Note that this algorithm never uses the value of $h(a)$, it just tries to compute $h(a)$. Furthermore, note that a is not an input to the algorithm. This is a scheme giving infinitely many algorithms. For each $a \in \mathbb{N}$, we have one algorithm $\texttt{ALG}_a$.

It is easy to see that $\texttt{ALG}_a$ computes the identity function if $a \in \mathcal{K}$. On the other hand, if $a \notin \mathcal{K}$, then $\texttt{ALG}_a$ will never terminate no matter what the input is. Hence, if $a \notin \mathcal{K}$, the algorithm computes the totally undefined function. The totally undefined function is definitely not the identity function (but it is a computable function). We conclude that $\texttt{ALG}_a$ computes the identity function if and only if $a \in \mathcal{K}$. Thus, $\texttt{ALG}_a$ computes a function which has an index that is an element of A if and only if $a \in \mathcal{K}$.

Let f_a denote the function computed by $\texttt{ALG}_a$. We know that f_a is either the identity function or the totally undefined function. Which of these two functions f_a is depends on the choice of a. To find a total computable function $\imath$ that satisfies (†), we need to realize that a computable index for the function f_a can be computed from the number a. Intuitively, this is not strange at all. We have a sequence of algorithms $\texttt{ALG}_0, \texttt{ALG}_1, \texttt{ALG}_2, \ldots$. The algorithms in the sequence are all very similar. The difference between $\texttt{ALG}_{17}$ and $\texttt{ALG}_2$ is that $\texttt{ALG}_{17}$ uses the number 17 in a place where $\texttt{ALG}_2$ uses the number 2. It is not hard to see that it should be possible to find an index for f_a if we know a. Indeed, there exists a primitive recursive function $\imath$ such that $\{\imath(a)\}(x) = f_a(x)$ (for all $x \in \mathbb{N}$). Thus, $\{\imath(a)\}$ is the identity function if $a \in \mathcal{K}$; and $\{\imath(a)\}$ is the totally undefined function if $a \notin \mathcal{K}$. Now, we can conclude that (*) holds since $\imath$ is a total computable function $\imath$ that satisfies (†). In a more formal and detailed proof, we need the S-m-n Theorem to conclude that the function $\imath$ exists, as you will see in a page or two.

With that introduction, you should be properly prepared to digest the proof of Rice's Theorem:

Proof. Assume that $\mathcal{A}$ is a nonempty set of computable functions that does not include every computable function, and let $A = \{e \mid \{e\} \in \mathcal{A}\}$. We need to prove that A is not a computable set.

As described above, the plan is to reduce the question of membership in the undecidable set $\mathcal{K}$ to membership in A. We will do this by constructing a total computable function $\imath : \mathbb{N} \to \mathbb{N}$ such that

$$a \in \mathcal{K} \quad \text{if and only if} \quad \imath(a) \in A. \tag{†}$$

Once we have this function, then if A were computable, $\mathcal{K}$ would be as well. Since we already know that $\mathcal{K}$ is not computable, we could conclude that A is not computable. So all we have to do is construct the function $\imath$ that satisfies condition (†).

Let $f_\uparrow(x)$ be the function that is undefined for any $x \in \mathbb{N}$. Now, $f_\uparrow$ is a computable function since $f_\uparrow(x) = (\mu i)[i + x < i]$. Since $\mathcal{A} \cup \overline{\mathcal{A}}$ contains all the computable functions, we either have $f_\uparrow \in \mathcal{A}$ or $f_\uparrow \in \overline{\mathcal{A}}$. Without loss of

generality, assume that $f_\uparrow \in \overline{\mathcal{A}}$. (If $f_\uparrow \in \mathcal{A}$, do a symmetric argument, and prove that $\overline{A}$ is not computable. The theorem follows as A is computable if and only if $\overline{A}$ is computable.)

The set $\mathcal{K}$ is semi-computable, and thus there is a computable function h such that $\mathcal{K} = \text{dom}(h)$.

As the set $\mathcal{A}$ is nonempty, fix a function $g \in \mathcal{A}$. Then define the function ϕ by

$$\phi(a, x) \;=\; (0 \;\dot{-}\; h(a)) + g(x) \;.$$

Obviously, ϕ is a computable function and, thus, ϕ has a computable index. Fix an index d such that $\{d\}(a, x) = \phi(a, x)$. By the S-m-n Theorem, we have a primitive recursive function S_1^1 such that $\{S_1^1(d, a)\}(x) = \{d\}(a, x)$. Let $\imath(a) = S_1^1(d, a)$. Then $\imath$ is a primitive recursive (and thus total computable) function. If we can show that $\imath$ satisfies the condition (†), we will be finished.

For the forward direction of (†), assume that $a \in \mathcal{K}$. This means that $h(a)$ is defined, as $\mathcal{K} = \text{dom}(h)$, and we have

$$\{\imath(a)\}(x) \;=\; \{S_1^1(d, a)\}(x) \;=\; \{d\}(a, x) \;=\; \phi(a, x) \;= \\ (0 \;\dot{-}\; h(a)) + g(x) \;=\; g(x)$$

and thus $\{\imath(a)\} \in \mathcal{A}$, which means that $\imath(a) \in A$, as needed.

Now we can address the reverse direction of (†). Assume that $a \notin \mathcal{K}$. Then $h(a)$ is not defined, and we have

$$\{\imath(a)\}(x) \;=\; \{S_1^1(d, a)\}(x) \;=\; \{d\}(a, x) \;=\; \phi(a, x) \;= \\ (0 \;\dot{-}\; h(a)) + g(x) \;=\; f_\uparrow(x)$$

and thus $\{\imath(a)\} \in \overline{\mathcal{A}}$ and $\imath(a) \notin A$, as wished.

Having produced a function $\imath$ that satisfies (†), we can finish the argument. For the sake of contradiction, assume that A is a computable set. Then, A has a computable characteristic function χ_A. Let $\chi(x) = \chi_A(\imath(x))$. By (†), we conclude that the total computable function χ is the characteristic function of $\mathcal{K}$. Thus, $\mathcal{K}$ is a computable set. This contradicts Theorem 7.6.12. □

7.8.1 Exercises

1. Let A be the set of all squares, that is, $A = \{0, 1, 4, 9, 16, \ldots\}$. Find a Diophantine equation $p(y, x_1, \ldots, x_n) = 0$ such that

   $$p(a, x_1, \ldots, x_n) = 0$$

   has a solution iff $a \in A$.

2. A natural number a is composite if there exist natural numbers $a, b > 1$ such that $a = bc$. Let A be the set of all composite natural numbers. Find a Diophantine equation $p(y, x_1, \ldots, x_n) = 0$ such that $p(a, x_1, \ldots, x_n) = 0$ has a solution iff $a \in A$.

3. Let A be the set of all natural numbers that are not a power of 2, that is
$$A = \{a \mid a \neq 2^x \text{ for all } x \in \mathbb{N}\} .$$
Find a Diophantine equation $p(y, x_1, \ldots, x_n) = 0$ such that
$$p(a, x_1, \ldots, x_n) = 0$$
has a solution iff $a \in A$. (This is a hard exercise.)

4. Use Rice's Theorem to prove that the set $\{e \mid \{e\}(17) = 314\}$ is not computable.

5. Let
$$A = \{e \mid \{e\}(x) = x \text{ for all } x \in \mathbb{N}\} \text{ and}$$
$$B = \{e \mid \text{for all } x \in \mathbb{N}, \text{ if } \{e\}(x) \text{ is defined, then } \{e\}(x) = x\} .$$
It follows from Rice's Theorem that A and B are not computable. Are these sets semi-decidable? What about the complement sets $\overline{A}$ and $\overline{B}$? Are these sets semi-decidable? Let $C = \{e \mid \{e\} \text{ is a total function}\}$. By Rice's Theorem, C is not a computable set. Is C a semi-decidable set? Is $\overline{C}$ a semi-decidable set?

6. (a) Use the MRDP Theorem to prove that there exists a Diophantine equation $p_0(u, y_1, \ldots, y_m) = 0$ (with integer coefficients) such that for any Diophantine equation $p(x_1, \ldots, x_n) = 0$ (with integer coefficients) there exists $k \in \mathbb{N}$ such that
$$p(x_1, \ldots, x_n) = 0 \text{ has a solution in } \mathbb{N} \quad \Leftrightarrow \quad p_0(k, y_1, \ldots, y_m) = 0 \text{ has a solution in } \mathbb{N} .$$

   (b) Let $p_0(u, y_1, \ldots, y_m) = 0$ be the equation given in (a). Give an informal explanation of why there exists an algorithm which given an arbitrary Diophantine equation $p(x_1, \ldots, x_n) = 0$ (with integer coefficients) can compute a natural number k such that
$$p(x_1, \ldots, x_n) = 0 \text{ has a solution in } \mathbb{N} \quad \Leftrightarrow \quad p_0(k, y_1, \ldots, y_m) = 0 \text{ has a solution in } \mathbb{N} .$$
   How will this algorithm work?

Chapter 8

Summing Up, Looking Ahead

There is more than one way to skin a cat. We're not quite sure why anyone would want to skin a cat, but, as Dickens almost says, the wisdom of our ancestors is in the saying; and our unhallowed hands shall not disturb it, or the Country's done for.

If you have worked through the entire book to this point, you have seen two different approaches to Gödel's Incompleteness Theorems, and those two approaches have given you a taste of two of the subfields of mathematical logic. Our first pass at the theorem, outlined in Chapters 5-6, was mostly a proof-theoretic approach. The story told in Chapter 7, on the other hand, has given you a nice introduction to the history and the basic results of computability theory, the branch of logic that is closer in feeling to theoretical computer science.

This chapter leads you through a review of the material that we have covered in the text, in a rather unusual form. We have prepared a series of exercises that are intended to revisit some of the major themes and results of the text, but to revisit them in a new context. Thus we'll begin by introducing you to bit strings and then we'll see what we can say about the language, models, and theory of these bit strings.

8.1 Once More, With Feeling

The best way to learn mathematics is by doing mathematics. To learn mathematics properly you have to do exercises and solve problem on your own. In the next couple of sections you will find a series of related exercises on material from all through the book. These exercises should help you to get a deeper understanding of what this book is all about. The results are pretty neat, too.

In some of the exercises below we will work with *strings* over alphabets. An *alphabet* is nothing but a set of symbols, while a string is a finite sequence of elements of the alphabet. If A is an alphabet it is customary to denote the set of all strings over the alphabet by $A^\star$. The set $A^\star$ always contains the empty string, ε. For example

$$\{a\}^\star \;=\; \{\varepsilon, a, aa, aaa, aaa, \ldots\}$$

and

$$\{\mathbf{0},\mathbf{1}\}^\star \;=\; \{\varepsilon, \mathbf{0}, \mathbf{1}, \mathbf{00}, \mathbf{01}, \mathbf{10}, \mathbf{11}, \mathbf{000}, \ldots\}\,.$$

The symbols in the alphabet $\{\mathbf{0},\mathbf{1}\}$ will be called *bits*, and we will refer to $\mathbf{0}$ and $\mathbf{1}$ as, respectively, *zero* and *one*. Note that the bits are written in boldface. This is done in order to distinguish bits and strings of bits from other things, like natural numbers.

If s is a string over an alphabet, then $|s|$ denotes the length of the string. So $|\mathbf{010}| = 3$ and $|\varepsilon| = 0$. If a is an alphabet symbol and $n \in \mathbb{N}$, then a^n denote the string that consists of n occurrences of a. Thus $a^5 = aaaaa$ and $a^0 = \varepsilon$.

8.2 The Language $\mathcal{L}_{BT}$ and the Structure $\mathfrak{B}$.

Let $\circ$ be a binary function symbol, and let 0, 1, and e be constant symbols. (Be careful here. The constant symbols are 0, 1, and e. Not $\mathbf{0}$ and $\mathbf{1}$, but 0 and 1.) Let $\mathcal{L}_{BT}$, the language of bit theory, be the first-order language $\{0, 1, e, \circ\}$,

8.2.1 Exercise

Write the following formulas in the official $\mathcal{L}_{BT}$ syntax, that is, use prefix notation in place of infix notation, and so on.

- $x \circ (y \circ z) = (x \circ y) \circ z$
- $(\forall x)(\forall y)(\forall z)\ x \circ (y \circ z) = (x \circ y) \circ z$
- $x \neq y \;\rightarrow\; (0 \circ x \neq 0 \circ y \wedge 1 \circ x \neq 1 \circ y)$

Let $\mathfrak{C}$ be the $\mathcal{L}_{BT}$-structure where

- the universe C is the set of natural numbers, that is, $C = \mathbb{N}$
- $e^{\mathfrak{C}} = 0$ and $0^{\mathfrak{C}} = 1^{\mathfrak{C}} = 1$ (yes, we interpret the constant symbol 0 as the number 1)
- $\circ^{\mathfrak{C}}$ is standard addition of natural numbers.

Let $\mathfrak{D}$ be the $\mathcal{L}_{BT}$-structure where

- the universe D is the set $\{a\}^\star$
- $e^{\mathfrak{D}} = \varepsilon$ and $0^{\mathfrak{D}} = 1^{\mathfrak{D}} = a$
- $\circ^{\mathfrak{D}}$ is concatenation of strings (we have, e.g., $aa \circ^{\mathfrak{D}} aaa = aaaaa$ and $\varepsilon \circ^{\mathfrak{D}} aaa = aaa$).

8.2.2 Exercise

Prove that $\mathfrak{C} \cong \mathfrak{D}$ ($\mathfrak{C}$ and $\mathfrak{D}$ are isomorphic).

The language $\mathcal{L}_{NT}$ has a standard interpretation, and an $\mathcal{L}_{NT}$-formula is normally read as a statement about natural numbers. There is a standard $\mathcal{L}_{NT}$-structure $\mathfrak{N}$ where the universe is the set of natural numbers. When an $\mathcal{L}_{NT}$-formula is interpreted in $\mathfrak{N}$, it becomes a statement about natural numbers. If we just say that an $\mathcal{L}_{NT}$ formula ϕ is true, we mean that ϕ is true in $\mathfrak{N}$. We will now introduce a corresponding standard structure $\mathfrak{B}$ for the language $\mathcal{L}_{BT}$. When an $\mathcal{L}_{BT}$-formula is interpreted in $\mathfrak{B}$, it becomes a statement about bit strings.

The universe of $\mathfrak{B}$ is the set of all *bit strings*, that is, the set $\{\mathbf{0}, \mathbf{1}\}^\star$. Furthermore, the constant symbol 0 is interpreted as the string containing nothing but the bit $\mathbf{0}$, and the constant 1 is interpreted as the string containing nothing but the bit $\mathbf{1}$, that is, $0^{\mathfrak{B}} = \mathbf{0}$ and $1^{\mathfrak{B}} = \mathbf{1}$. The constant e is interpreted as the empty string, that is, $e^{\mathfrak{B}} = \varepsilon$. Finally, $\circ^{\mathfrak{B}}$ is the function that concatenates two strings (e.g. $\mathbf{01} \circ^{\mathfrak{B}} \mathbf{001} = \mathbf{01001}$ and $\varepsilon \circ^{\mathfrak{B}} \varepsilon = \varepsilon$).

8.2.3 Exercise

Prove that $\mathfrak{B} \ncong \mathfrak{C}$ ($\mathfrak{B}$ is not isomorphic to $\mathfrak{C}$).

The $\mathcal{L}_{NT}$-terms $\overline{0}, \overline{1}, \overline{2}, \ldots$ are called numerals. We can define the numerals inductively: $\overline{0} = 0$ and $\overline{n+1} = S\overline{n}$. The numerals name all of the elements in the standard structure: for each $n \in \mathbb{N}$ there exists a numeral $\overline{n}$ such that $\overline{n}^{\mathfrak{N}} = n$. We will now introduce corresponding terms for the standard $\mathcal{L}_{BT}$-structure $\mathfrak{B}$ – let us call these terms *biterals*.

Let $b \in \{\mathbf{0}, \mathbf{1}\}^\star$. We define the biteral $\overline{b}$ by using recursion on the length of b:

- $\overline{\varepsilon} = e$
- $\overline{\mathbf{0}b} = (0 \circ \overline{b})$
- $\overline{\mathbf{1}b} = (1 \circ \overline{b})$.

Every element in the universe of the standard $\mathcal{L}_{BT}$-structure $\mathfrak{B}$ is named by a unique biteral, and we have, e.g., $\overline{\mathbf{101}} = (1 \circ (0 \circ (1 \circ e)))$. Note that, e.g., $((1 \circ 0) \circ (1 \circ e))$ is not a biteral.

Let $\mathit{init}(x, y)$ be the $\mathcal{L}_{BT}$-formula $(\exists z) x \circ z = y$. Then, $\mathit{init}(x, y)$ states that the bit string x is an initial segment of the bit string y, that is, for any $b_1, b_2 \in \{\mathbf{0}, \mathbf{1}\}^\star$, we have

$$\mathfrak{B} \models \mathit{init}(\overline{b_1}, \overline{b_2}) \quad \text{if and only if} \quad b_1 \text{ is an initial segment of } b_2.$$

Similarly, the formula

$$\mathit{end}(x, y) :\equiv (\exists z) z \circ x = y$$

states that x is an end segment of y. The formula

$$\mathit{notzz}(x) :\equiv \neg(\exists y)(\exists z)\,(x = y \circ 0 \circ 0 \circ z)$$

states that the bit string x does not contain the substring $\mathbf{00}$. Since the function $\circ$ is associative (in the standard structure $\mathfrak{B}$), we do not bother to write parentheses in expressions like $y \circ 0 \circ 0 \circ z$ (when we want the reader to interpret these expressions in the standard structure).

8.2.4 Exercise

- Write up a statement $\mathit{ones}(x)$ which states that x contains no zeros.
- Write up a statement $\mathit{sub}(x, y)$ which states that x is a substring of y.
- Write up a statement $\phi(x, y)$ which states that every substring of x that contains only ones is also a substring of y.

At first glance, the language $\mathcal{L}_{BT}$ does not seem to be very expressive. There is only one function symbol in the language – there are no relation symbols. For example, it might not seem possible to state that there are more zeros than ones in a string, or that the number of bits in a string is even. It may not even seem possible to state that two strings are of the same length. Thus, it might be tempting to conclude right away that interesting mathematical assertions cannot be expressed by $\mathcal{L}_{BT}$-formulas. But we should think twice. The next few exercises show that $\mathcal{L}_{BT}$ is a surprisingly expressive language.

8.2.5 Exercise

Write an $\mathcal{L}_{BT}$-formula $\mathit{lessthan}(x, y)$ such that $\mathfrak{B} \models \mathit{lessthan}(\overline{a}, \overline{b})$ if and only if $a, b \in \{\mathbf{1}\}^\star$ and $|a| \leq |b|$. Write an $\mathcal{L}_{BT}$-formula $\mathit{slessthan}(x, y)$ such that $\mathfrak{B} \models \mathit{slessthan}(\overline{a}, \overline{b})$ if and only if $a, b \in \{\mathbf{1}\}^\star$ and $|a| < |b|$.

8.2.6 Exercise

Write up an $\mathcal{L}_{BT}$-formula $\mathit{even}(x)$ such that $\mathfrak{B} \models \mathit{even}(\overline{b})$ iff $b \in \{\mathbf{1}\}^\star$ and $|b|$ is even.

Suggestion: If you solved Exercise 8.2.5 without too much effort, you might want to look at our provided solutions to that problem before attacking this one.

8.2.7 Exercise

Write up an $\mathcal{L}_{BT}$-formula $\mathit{mult}(x, y, z)$ such that $\mathfrak{B} \models \mathit{mult}(\overline{a}, \overline{b}, \overline{c})$ iff $a, b, c \in \{\mathbf{1}\}^\star$ and $|a| \cdot |b| = |c|$.

8.2.8 Exercise

Write up a $\mathcal{L}_{BT}$-formula $\mathit{prime}(x)$ such that $\mathfrak{B} \models \mathit{prime}(\overline{b})$ iff $b \in \{\mathbf{1}\}^\star$ and $|b|$ is prime.

8.2.9 Exercise

The Goldbach Conjecture says that any even number strictly greater than two can be written as the sum of two primes. Write up an $\mathcal{L}_{BT}$-formula ϕ such that $\mathfrak{B} \models \phi$ iff the Goldbach Conjecture holds.

The preceding exercises shows that $\mathcal{L}_{BT}$ is a rather expressive language after all. Indeed, for each $\mathcal{L}_{NT}$-formula ϕ, we can construct an $\mathcal{L}_{BT}$-formula ϕ' such that $\mathfrak{N} \models \phi$ iff $\mathfrak{B} \models \phi'$. The converse is also true: for each $\mathcal{L}_{BT}$-formula ϕ', we can construct an $\mathcal{L}_{NT}$-formula ϕ such that $\mathfrak{B} \models \phi'$ iff $\mathfrak{N} \models \phi$. Thus, first-order bit theory and first order number theory capture the same fragment of mathematics.

Let us round off this section with a couple of exercises on how to encode sequences of natural numbers in first-order bit theory. We say that the bit string b encodes the nonempty sequence $(a_1, \ldots, a_n)$ of natural numbers if

$$b \;=\; \mathbf{00101}^{a_1+1}\mathbf{001}^{2}\mathbf{01}^{a_2+1}\mathbf{001}^{3}\mathbf{01}^{a_3+1}\mathbf{00}\ldots\mathbf{001}^{n}\mathbf{01}^{a_n+1}\mathbf{00}\,.$$

8.2.10 Exercise

Write up a an $\mathcal{L}_{BT}$-formula *code*(x) which states that x encodes a nonempty sequence of natural numbers, that is,

$$\mathfrak{B} \models \mathit{code}(\overline{b}) \quad \Leftrightarrow \quad b \text{ encodes } (a_1, \ldots, a_n)$$

for some $a_1, \ldots, a_n \in \mathbb{N}$ with $n \geq 1$.

In a later exercise you will need an $\mathcal{L}_{BT}$-formula *IthElement*(x, y, z) such that $\mathfrak{B} \models \mathit{IthElement}(\overline{\mathbf{1}^m}, \overline{\mathbf{1}^i}, \overline{b})$ iff

- b encodes some nonempty sequence $(a_1, \ldots, a_n)$ of naturals, and
- $1 \leq i \leq n$, and
- $m = a_i$.

Intuitively *IthElement*(x, y, z) says that if x is the bit string $\mathbf{1}^m$ and y is the bit string $\mathbf{1}^i$, then b represents a sequence of naturals $(a_1, \ldots, a_n)$ where $m = a_i$. (It does not matter what the formula says when x or y contain zeroes.)

8.2.11 Exercise

Write up the formula *IthElement*(x, y, z).

8.3 Nonstandard $\mathcal{L}_{BT}$-structures

Recall the nonstandard structures of number theory—the structures that contain strange numbers that are greater than any natural number. There will also be similar nonstandard models for bit theory. We say that a $\mathfrak{B}^*$ is a *nonstandard* $\mathcal{L}_{BT}$-structure when

- $\mathfrak{B}^*$ is elementarily equivalent to the standard structure ($\mathfrak{B}^* \equiv \mathfrak{B}$)
- $\mathfrak{B}^*$ is not isomorphic to the standard structure ($\mathfrak{B}^* \ncong \mathfrak{B}$)

8.3.1 Exercise

Prove that there exists a nonstandard $\mathcal{L}_{BT}$-structure.

Let $\mathfrak{B}^*$ be a nonstandard $\mathcal{L}_{BT}$-structure. We say that an element a in the universe of $\mathfrak{B}^*$ is a *standard* element if $a = \overline{b}^{\mathfrak{B}^*}$ for some $b \in \{\mathbf{0}, \mathbf{1}\}^\star$. The elements in the universe of $\mathfrak{B}^*$ that are not standard elements will be referred to as *nonstandard* elements.

8.3.2 Exercise

Let $\mathfrak{B}^*$ be any nonstandard $\mathcal{L}_{BT}$-structure. Prove that there are infinitely many nonstandard elements in the universe of $\mathfrak{B}^*$.

8.4 The Axioms of B

We will now introduce the set B of $\mathcal{L}_{BT}$-axioms. Thereafter, we will investigate what we can—and what we cannot—deduce from these axioms.

The Axioms of B

1. $(\forall x)x = e \circ x$.
2. $(\forall x)x = x \circ e$.
3. $(\forall x)(\forall y)(\forall z)(x \circ y) \circ z = x \circ (y \circ z)$.
4. $(\forall x)[\,0 \circ x \neq e \ \wedge\ 1 \circ x \neq e\,]$.
5. $(\forall x)(\forall y)0 \circ x \neq 1 \circ y$.
6. $(\forall x)(\forall y)[\,x \neq y \ \rightarrow\ (0 \circ x \neq 0 \circ y \ \wedge\ 1 \circ x \neq 1 \circ y)\,]$.

8.4.1 Exercise
Explain briefly why $B \vdash \phi$ entails $\mathfrak{B} \models \phi$.

8.4.2 Exercise
Give a B-deduction of $1 \circ e \neq 0 \circ e$. Provide the details. Name the inference rules, the logical axioms, and the nonlogical axioms involved in the deduction.

8.4.3 Exercise
Which (nonlogical) axioms of B do you need to deduce $0 \neq 1$?

8.4.4 Exercise
Which (nonlogical) axioms of B do you need to deduce $1 \circ 1 \neq 1$?

8.4.5 Exercise
Which (nonlogical) axioms of B do you need to deduce $1 \circ 1 = 1 \circ 1$?

8.4.6 Exercise
Which (nonlogical) axioms of B do you need to deduce $1 \circ 1 \circ 1 = 1 \circ 1 \circ 1$?

8.4.7 Exercise
Let $y_n, \ldots, y_1$ be variables, and let t be an $\mathcal{L}_{BT}$-term. Prove that

$$B \vdash (y_n \circ (y_{n-1} \circ \ldots (y_1 \circ e) \ldots)) \circ t \;=\; (y_n \circ (y_{n-1} \circ \ldots (y_1 \circ t) \ldots))$$

Use induction on n.

8.4.8 Exercise
Prove that for any variable-free $\mathcal{L}_{BT}$-term t there exists a biteral b such that $B \vdash t = b$.

8.4.9 Exercise
Let t_1 and t_2 be any variable-free $\mathcal{L}_{BT}$-terms. Prove that

$$\mathfrak{B} \models t_1 = t_2 \;\Rightarrow\; B \vdash t_1 = t_2 \,.$$

So far we have not needed to use axioms B4 or B5. So the three first axioms of B are sufficient to deduce that $t_1 = t_2$ whenever t_1 and t_2 are variable-free $\mathcal{L}_{BT}$-terms such that $\mathfrak{B} \models t_1 = t_2$. In the next few exercises you are asked to prove that we can deduce $t_1 \neq t_2$ whenever t_1 and t_2 are variable-free $\mathcal{L}_{BT}$-terms such that $\mathfrak{B} \models t_1 \neq t_2$. Now you will need B4, B5, and B6. These three axioms are related to the two first axioms of N:

- $(\forall x)Sx \neq 0$
- $(\forall x)(\forall y)[Sx = Sy \rightarrow x = y]$ (Notice that this is logically equivalent to $(\forall x)(\forall y)[x \neq y \rightarrow Sx \neq Sy]$).

The $\mathcal{L}_{NT}$-formulas N1 and N2 are the nonlogical axioms you need to deduce $\overline{m} \neq \overline{n}$ for two natural numbers m and n where $n \neq m$. The $\mathcal{L}_{BT}$-formulas B4, B5, and B6 are the non-logical axioms you need to deduce $\overline{a} \neq \overline{b}$ for two bit strings a and b where $a \neq b$.

8.4.10 Exercise

Let b_1 and b_2 be biterals. Prove that

$$\mathfrak{B} \models b_1 \neq b_2 \quad \Rightarrow \quad B \vdash b_1 \neq b_2 \, .$$

8.4.11 Exercise

Let t_1 and t_2 be variable-free $\mathcal{L}_{BT}$-terms. Prove that

$$\mathfrak{B} \models t_1 \neq t_2 \quad \Rightarrow \quad B \vdash t_1 \neq t_2 \, .$$

You certainly remember that, when we were working in the language of number theory, we proved that we can deduce any true-in-$\mathfrak{N}$ Σ-sentence from the axioms of N. Can we prove a similar result for the axioms of B when we are working in $\mathcal{L}_{BT}$? One complication is that bounded quantifiers are not available in the language $\mathcal{L}_{BT}$. This means that we cannot easily define a class of $\mathcal{L}_{BT}$-formulas that corresponds to the class of Σ-formulas. It should be possible to introduce some kind of bounded quantifiers in the language $\mathcal{L}_{BT}$, but we will not undertake that project here. Instead we will define what it means for an $\mathcal{L}_{BT}$-formula to be an *existential sentence*. It turns out that we can deduce all true existential sentences from the axioms of B.

We define an *existential* first-order formula inductively:

(i) ϕ is an existential formula if ϕ is an atomic formula

(ii) $\neg\phi$ is an existential formula if ϕ is an atomic formula

(iii) $(\alpha \wedge \beta)$ is an existential formula if α and β are existential formulas

(iv) $(\alpha \vee \beta)$ is an existential formula if α and β are existential formulas

(v) $(\exists x)(\phi)$ is an existential formula if ϕ is an existential formula.

An *existential sentence* is an existential formula with no free variables.

8.4.12 Exercise

Let ϕ be an existential sentence in the language of $\mathcal{L}_{BT}$ that is true in standard structure $\mathfrak{B}$. Prove that $B \vdash \phi$.

8.5 B extended with an induction scheme

Consider the $\mathcal{L}_{BT}$-sentence

$$(\forall x)\big[\, x \neq e \;\; \rightarrow \;\; (\exists y)[0 \circ y = x \;\vee 1 \circ y = x]\,\big] \;.$$

The sentence is definitely true in the standard model: Every nonempty bit string is the result of concatenating **0**, or concatenating **1**, with another bit string. Can we deduce this sentence from the axioms of B? We know that we can deduce every true existential sentence from the axioms of B, but this is not an existential sentence. There is a universal quantifier there.

There is an obvious way to prove that a formula is deducible from a set of axioms: provide a deduction. Alternatively, we could argue that the formula follows logically from the axioms, and then use the Completeness Theorem to conclude that the formula is deducible. But how can we prove that a formula ϕ is not deducible from the axioms of B? Well, the standard method is to provide a model $\mathfrak{A}$ for B such that $\mathfrak{A} \not\models \phi$.

8.5.1 Exercise

Explain briefly why $\mathfrak{A} \models B$ and $\mathfrak{A} \not\models \phi$ implies $B \nvdash \phi$.

8.5.2 Exercise

Prove that

$$B \nvdash (\forall x)\big[\, x \neq e \;\; \rightarrow \;\; (\exists y)[0 \circ y = x \;\vee 1 \circ y = x]\,\big] \;.$$

Let

$$B_1 \;=\; B \;\cup\; \{(\forall x)\big[\, x \neq e \;\rightarrow\; (\exists y)[0 \circ y = x \;\vee 1 \circ y = x]\,\big]\}$$

and let

$$B_2 \;=\; B \;\cup\; \{\neg(\forall x)\big[\, x \neq e \;\rightarrow\; (\exists y)[0 \circ y = x \;\vee 1 \circ y = x]\,\big]\}\,.$$

Recall that a set of axioms A is consistent if $A \nvdash \bot$ (where $\bot$ is an abbreviation for the formula $[(\forall x)x = x] \wedge \neg[(\forall x)x = x]$).

8.5.3 Exercise

Explain briefly why both B_1 and B_2 are consistent set of axioms.

The next exercise is hard. Consider yourself warned.

8.5.4 Exercise

Prove that $B \nvdash (\forall x)0 \circ x \neq x$.

You will recall from Section 6.6 that the axioms of PA (Peano Arithmetic) consist of the eleven axioms of N together with an infinite number of axioms given by the *induction scheme*:

$$\big(\; \phi(0) \;\wedge\; (\forall x)[\,\phi(x) \;\rightarrow\; \phi(S(x))\,]\;\big) \;\rightarrow\; (\forall x)\phi(x)\,.$$

We have an axiom of this form for each $\mathcal{L}_{NT}$-formula ϕ with one free variable. From the axioms of PA we can deduce a number of natural Π-formulas that cannot be proven from the axioms of N alone. For example, PA is strong enough to prove that addition is commutative, that is, $PA \vdash (\forall x)(\forall y)x + y = y + x$. One of our early uses of nonstandard models of arithmetic was to show that this formula cannot be deduced from the axioms N. In general, PA, although not complete, is far more powerful than the N with respect to proving Π-formulas.

Using a similar idea, we can strengthen B by adopting a first order induction scheme for bit strings:

$$\big(\; \phi(e) \;\wedge\; (\forall x)[\,\phi(x) \;\rightarrow\; (\phi(0 \circ x) \;\wedge\; \phi(1 \circ x))\,]\;\big) \;\rightarrow\; (\forall x)\phi(x)\,.$$

Let B_I be the collection of axioms that consists of the axioms B together with the axioms given by the induction scheme above. It is obvious that $\mathfrak{B} \models B_I$.

8.5.5 Exercise
Prove that $B_I \vdash (\forall x)0 \circ x \neq x$.

It is also the case that

$$B_I \vdash (\forall x)\big[\, x \neq e \;\; \rightarrow \;\; (\exists y)[0 \circ y = x \;\vee 1 \circ y = x]\,\big] \; .$$

Indeed, it is very hard to find true natural sentences that cannot be deduced from the axioms of B_I. So how powerful are the axioms of B_I? Can every sentence true in $\mathfrak{B}$ be deduced from the axioms in B_I? That seems unlikely, right? Since the exercises in Section 8.2 indicate that a fair amount of mathematics can be expressed in the language $\mathcal{L}_{BT}$ and proved from the axioms in B, we ought to believe that the sort of arguments that worked in order to show that N was not a complete set of axioms should also work in the setting of bit strings. Thus, we should expect that the axioms of B_I are incomplete.

8.6 Incompleteness

We say that an $\mathcal{L}_{BT}$-formula $\phi(x_1, \ldots, x_n, y)$ *defines* a function $f : \mathbb{N}^n \to \mathbb{N}$ when

$$\mathfrak{B} \models \phi(\overline{\mathbf{1}^{a_1}}, \ldots, \overline{\mathbf{1}^{a_n}}, \overline{\mathbf{1}^{b}}) \quad \Leftrightarrow \quad f(a_1, \ldots, a_n) = b \; .$$

8.6.1 Exercise
Prove that any computable function can be defined by an $\mathcal{L}_{BT}$-formula.

Hint: Use induction on the structure of the computable function f. Thus, you prove that such a formula exists when f is one of the initial functions (zero, successor, projection); when f is defined by the composition scheme; when f is defined by the scheme of primitive recursion; when f is defined by minimalization. In the case when f is defined by primitive recursion, you will need the formula *IthElement*(x, y, z).

Recall the set $\mathcal{K}$ that was defined by $\mathcal{K} = \{x \mid x \in \mathcal{W}_x\}$ where $\mathcal{W}_i$ denotes the i^{th} semi-computable set. The set $\mathcal{K}$ is semi-computable, but not computable. The complementary set of $\mathcal{K}$, the set $\overline{\mathcal{K}}$, is not semi-computable.

8.6.2 Exercise
Prove that there exists an $\mathcal{L}_{BT}$-formula $\phi(x)$ such that $\mathfrak{B} \models \phi(\overline{\mathbf{1}^{a}})$ if and only if $a \in \mathcal{K}$.

Symbol	Symbol Number	Symbol	Symbol Number
$\neg$	1	e	9
$\vee$	3	0	11
$\forall$	5	1	13
$=$	7	$\circ$	15
		v_i	$2i$

Table 8.1: Symbol Numbers for $\mathcal{L}_{BT}$

We use the enumeration of symbols in Table 8.1 to assign Gödel numbers to the $\mathcal{L}_{BT}$-formulas. We assign the Gödel number $\ulcorner t \urcorner$ to the term t by

$$\ulcorner t \urcorner = \begin{cases} 2^9 & \text{if } t \text{ is } e \\ 2^{11} & \text{if } t \text{ is } 0 \\ 2^{13} & \text{if } t \text{ is } 1 \\ 2^{15} 3^{\ulcorner t_1 \urcorner} 5^{\ulcorner t_2 \urcorner} & \text{if } t \text{ is } \circ t_1 t_2 \\ 2^{2i} & \text{if } t \text{ is the variable } v_i \end{cases}$$

and we assign the Gödel number $\ulcorner \phi \urcorner$ to the formula ϕ by

$$\ulcorner \phi \urcorner = \begin{cases} 2^1 3^{\ulcorner \alpha \urcorner} & \text{if } s \text{ is } (\neg \alpha) \\ 2^3 3^{\ulcorner \alpha \urcorner} 5^{\ulcorner \beta \urcorner} & \text{if } s \text{ is } (\alpha \vee \beta) \\ 2^5 3^{\ulcorner v_i \urcorner} 5^{\ulcorner \alpha \urcorner} & \text{if } s \text{ is } (\forall v_i)(\alpha) \\ 2^7 3^{\ulcorner t_1 \urcorner} 5^{\ulcorner t_2 \urcorner} & \text{if } s \text{ is } = t_1 t_2 \end{cases}$$

Now, we have assigned a unique number Gödel number $\ulcorner \phi \urcorner$ to each $\mathcal{L}_{BT}$ formula ϕ. (No, this is not the same assignment of Gödel numbers that we used for $\mathcal{L}_{NT}$ in Section 5.7. Use this definition of $\ulcorner \phi \urcorner$ for the exercises that follow.)

8.6.3 Exercise

Let $\phi(x)$ be an $\mathcal{L}_{BT}$-formula, and let $f_\phi : \mathbb{N} \to \mathbb{N}$ be the function given by $f_\phi(a) = \ulcorner \phi(\overline{\mathbf{1}^a}) \urcorner$. Explain why f_ϕ is a primitive recursive function.

8.6.4 Exercise

Let $\phi(x)$ be an $\mathcal{L}_{BT}$-formula, and let A be a set of $\mathcal{L}_{BT}$-axioms such that the set $\{\ulcorner \eta \urcorner \mid A \vdash \eta\}$ is semi-computable. Prove that the set $\{a \mid A \vdash \phi(\overline{\mathbf{1}^a})\}$ is semi-computable.

8.6.5 Exercise
Let A be a set of $\mathcal{L}_{BT}$-axioms such that the set $\{\ulcorner\eta\urcorner \mid A \vdash \eta\}$ is semi-computable and $\mathfrak{B} \models A$. Prove that there is an $\mathcal{L}_{BT}$-sentence θ such that $\mathfrak{B} \models \theta$ but $A \nvdash \theta$.

8.6.6 Exercise
Explain briefly why there exists an $\mathcal{L}_{BT}$-formula θ such that $\mathfrak{B} \models \theta$ and $B_I \nvdash \theta$.

8.6.7 Exercise
Explain briefly why there exists an $\mathcal{L}_{BT}$-formula θ such that $B_I \nvdash \theta$ and $B_I \nvdash \neg\theta$.

8.7 Off You Go

Well, that was pretty cool, wasn't it? We showed that the language $\mathcal{L}_{BT}$, even though it has only a single binary function symbol and three constant symbols, is rich enough to allow us to express enough number theory to do coding, and through that coding we found that B_I is incomplete. It just seems that incompleteness is a natural result and rather hard to avoid. Which is probably not at all what Hilbert expected when he brought the problem up.

We hope that you have found this tour of some of the basic results of mathematical logic enlightening and informative. As you can imagine, the field of logic is much broader and much deeper than we have had time, space, (or inclination) to cover in this text. From multivalued logic to Boolean algebra, from set theory to model theory, vast areas of research and fundamental results have been either glossed over or totally ignored. But we do hope that we have given you a glimpse of a truly fascinating field.

We believe that logic is uniquely placed at the intersection of mathematics, philosophy, and computer science. From this prime piece of real estate, we can appreciate the interactions of these three fields of intellectual endeavor and understand a bit more about the strengths and limitations of these approaches to understanding ourselves and our world. It is fascinating stuff, and we hope that you have enjoyed the journey.

C.L. and L.K.
Geneseo and Oslo
July 2015

Appendix
Just Enough Set Theory to Be Dangerous

It is our goal in this Appendix to review some basic set-theoretic notions and to state some results that are used in the book. Very little will be proved, but the exercises will give you a chance to get a feel for the subject. There are several laudable texts and reference works on set theory: we have listed a few in the Bibliography.

Think of a set as a collection of objects. If X is a set, we write $a \in X$ to say that the object a is in the collection X. For our purposes, it will be necessary that any given thing either is an element of a given set or is not an element of that set. If that sounds obvious to you, suppose we asked you whether or not 153,297 was in the "set" of big numbers. Depending on your age and whether or not you are used to handling numbers that large, your answer might be "yes," "no," or "pretty large, but not all that large." So you can see that membership in an alleged set might not be all that cut and dried. But we won't think about sets of that sort, leaving them to the field of fuzzy set theory.

Our main concern will be with infinite sets and the size of those sets. One of the really neat results of set theory is that infinite sets come in different sizes, so there are different sizes of infinity! Here are some of the details: We say that sets X and Y have the same **cardinality**, and write $|X| = |Y|$, if there is a one-to-one and onto function (also called a *bijection*) $f : X \to Y$. The function f is sometimes called a *one-to-one correspondence.* For example, if X and Y are finite sets, this just means that you can match up the elements of the set without missing anyone. So if $D = \{\text{Happy, Sleepy, Grumpy, Doc, Dopey, Sneezy, Bashful}\}$ and $W = \{\text{Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, Saturday}\}$, it is easy to see that $|D| = |W|$ by pairing them up. Notice in doing the pairing that you never actually have to count the number of elements in either set. All you have to do is match them up. This is what allows us to apply the concept of cardinality to infinite sets.

As an easy example, notice that $\mathbb{N} = \{0, 1, 2, 3, \dots\}$ has the same cardinality as $\textsc{Even} = \{0, 2, 4, 6, \dots\}$. To prove this, we must exhibit a bijection between the two sets, and the function $f : \mathbb{N} \to \textsc{Even}$ defined by $f(x) = 2x$ works quite nicely. It is easy to check that f is a bijection, so $|\textsc{Even}| = |\mathbb{N}|$. Notice that this says that these two sets have the same size, even though the first is a proper subset of the second. This cannot happen with finite sets, but is rather common with infinite sets.

A set that has the same cardinality as the set of natural numbers is called **countable**. The exercises will give you several opportunities to show that the cardinality of one set is equal to the cardinality of another set.

We will say that the cardinality of set A is less than or equal to the cardinality of set B, and write $|A| \leq |B|$, if there is a one-to-one function $f : A \to B$. If $|A| \leq |B|$ but $|A| \neq |B|$, we say that the cardinality of A is less than the cardinality of B and write $|A| < |B|$. One of the basic theorems of set theory, the Schröder–Bernstein Theorem, says that if $|A| \leq |B|$ and $|B| \leq |A|$, then $|A| = |B|$. The proof of this theorem is beyond this little Appendix, but a nice version is in [Hrbacek and Jech 84]. Be warned, however. This theorem looks obvious, because of the notation. What it says, however, is that if you have two sets A and B such that there are injections $f_1 : A \to B$ and $f_2 : B \to A$, then there is a bijection $f_3 : A \to B$. But for our purposes, it will suffice to think: "If the size of this set is no bigger than the size of that set, and if the size of that set is no bigger than the size of this set, then the two sets must have the same size!"

With this tool in hand it is not hard to show that $\mathbb{Q}$, the collection of rational numbers, is countable. An injection from $\mathbb{N}$ to $\mathbb{Q}$ is easy: $f(x) = x$ will do. But to find an injection from the rationals to the naturals requires more thought. We will let you think about it, but trust us when we say that there is such a function. If you don't trust me, check any introductory set theory book or discrete mathematics textbook.

You will have noticed by this point that all we have done is show that several different infinite sets are countable. Georg Cantor proved in 1874 that the set of real numbers is not countable, in other words, he showed that $|\mathbb{N}| < |\mathbb{R}|$. In 1891 he developed his famous diagonal argument to show that given any set X, there is a set of larger cardinality, namely the collection of all subsets of X, which we call the *power set* of X and denote $P(X)$. So Cantor's Theorem states that $|X| < |P(X)|$. Thus there can be no largest cardinality. More picturesquely: There is no largest set.

The way to think about a countable set is that you can write an infinite list that contains every element of the set. This means that you can go through the set one element at a time and be sure that you eventually get to every element of the set, and that any element of the set only has finitely many things preceding it on the list. Think about listing the natural numbers. Although it would take a long time to reach the number 1038756397456 on the list $(0, 1, 2, 3, 4, \dots)$, we would eventually get to that

number, and after only a finite amount of time. The fact that the real numbers are uncountable means that if you try to write them out in a list, you have to leave some reals off your list. So if you try to list the reals as $\langle 1, -4/7, e, 10^{10}, \pi/4, \ldots\rangle$, then there has to be a real number, maybe 17, that you left out.

Another nice fact about countable sets: Suppose that X_1, X_2, X_3, ... are all countable sets, and consider the set $Y = X_1 \cup X_2 \cup X_3 \cup \cdots$. So Y is a countable union of countable sets. If you want to know the size of Y, it is easy to find, for there is a theorem that states that the countable union of countable sets is countable. (Purists: Calm down. We know about the Axiom of Choice. We know we don't need this theorem in this generality. But this isn't a set theory text, so lighten up a little!) Let's look at an application of this wonderful theorem: Almost all of the languages in this book are countable, meaning that the set consisting of all of the constant symbols, function symbols, and relation symbols, together with the connectives, parentheses, and variables is countable. Now using the fact that a countable union of countable sets is countable, we can show that S_2, the collection of all strings of two symbols from the language, is countable. Here is a table that contains every string of exactly two $\mathcal{L}$-symbols, where $\mathcal{L} = \{s_0, s_1, s_2, \ldots\}$:

	s_0	s_1	s_2	s_3	s_4	$\cdots$
s_0	s_0s_0	s_0s_1	s_0s_2	s_0s_3	s_0s_4	
s_1	s_1s_0	s_1s_1	s_1s_2	s_1s_3	s_1s_4	$\cdots$
s_2	s_2s_0	s_2s_1	s_2s_2	s_2s_3	s_2s_4	
s_3	s_3s_0	s_3s_1	s_3s_2	s_3s_3	s_3s_4	
			$\vdots$			

Now, this array shows that S_2 can be thought of as a countable union of countable sets: Each row of the table is countable, and there are countably many rows. By the theorem of the last paragraph, the collection of length-two strings is countable.

> *Chaff:* Notice that by reading the table diagonally starting in the upper left-hand corner, we can explicitly construct a listing of all of the length-two strings:
>
> $$(s_0s_0, s_0s_1, s_1s_0, s_0s_2, s_1s_1, s_2s_0, s_0s_3, s_1s_2, s_2s_1, s_3s_0, \ldots).$$
>
> This trick is important when you want to be able to program a computer to produce the listing or when you want to show that the collection of length two strings is computable.

Using the same idea, we can prove by induction that for any natural number n, S_n, the collection of strings of length n is countable. [Put the length $(n-1)$ strings across the top of the table and the symbols of $\mathcal{L}$ down

the side.] But then the collection of *all* finite strings is just the (countable) union $S_0 \cup S_1 \cup S_2 \cup \cdots$. So the collection of all finite strings is countable. Exercise 5 asks you to think about generating the listing of strings via a computer.

Exercises

1. Show that the set of numbers that are a nonnegative power of 10 (also known as $A = \{1, 10, 100, 1000, \dots\}$) is countable.

2. Show that the collection of prime numbers is countable.

3. (a) Show that these two intervals on the real line have the same cardinality: $[0, 1]$ and $[0, 10]$.

   (b) Show that these two intervals have the same cardinality: $[0, 1]$ and $[1, 3]$.

   (c) Show that these two intervals on the real line have the same cardinality: $[a, b]$ and $[c, d]$, where $a \neq b$ and $c \neq d$.

   (d) Show that the interval $[0, 1]$ and the interval $(0, 1)$ have the same cardinality. [*Suggestion:* The easy way is to quote a theorem. The interesting way is to find an explicit bijection between the two sets.]

4. Show that the relation "has the same cardinality as" is an equivalence relation.

5. Suppose that we have a recursively enumerable listing of the elements of a countable language $\mathcal{L} = (s_0, s_1, s_2, \dots)$. Design a computer program that lists all of the finite strings of symbols from the language, showing that the collection of finite strings is also recursively enumerable.

Solutions to Selected Exercises

Section 1.3.1

Section 1.3.1, Exercise 3, page 13:

For any natural number n, let $\overline{n}$ be an abbreviation for $\overbrace{SSS\ldots SS}^{n \text{ times}}0$, and let

$$Prime(x) \quad :\equiv \quad \overline{1} < x \ \wedge \ \neg(\exists y)(\exists z)[(y + \overline{2}) \cdot (z + \overline{2}) = x] \ .$$

When we read the symbols with the intended meaning, the formula $Prime(\overline{n})$ states that n is a prime number. The formula

$$(\forall x)(\exists y)[x < y \ \wedge \ Prime(y)]$$

states that there is no largest prime. The formula

$$(\forall x)(\exists y)[x < y \ \wedge \ Prime(y) \ \wedge \ Prime(y + \overline{2})]$$

states that there are infinitely many twin primes. So does the formula

$$(\forall x)(\exists y)(\exists z)[x < y \ \wedge \ y < z \ \wedge \ Prime(y) \ \wedge \ Prime(z) \ \wedge \ z < y + \overline{3}]$$

We do not know if there exists infinitely many twin primes, neither do we know if the formula

$$(\forall x)[\ (\exists y)[(y + \overline{2}) \cdot \overline{2} = x] \quad \rightarrow \\ (\exists u)(\exists v)[Prime(u) \ \wedge \ Prime(v) \ \wedge \ u + v = x]\ ] \ .$$

is true. The formula states the Goldbach Conjecture: *Any even number, except 0 and 2, can be written as a sum of two primes.*

The formula

$$(\forall x)(\exists y)(\exists z)\,[x < y \ \wedge \ y < z \ \wedge \ Prime(y) \ \wedge \ Prime(z) \ \wedge \\ z < y + \overline{70000000}] \ .$$

states the Bounded Gap Theorem. Compare this formula to the formula above stating that there are infinitely many twin primes. The Bounded Gap Theorem was proved by Yitang Zhang in 2013.

Section 1.4.1

Section 1.4.1, Exercise 1, page 17:
For any $n \in \mathbb{N}$, we have

$$0^2 + 1^2 + \cdots + n^2 \;=\; \frac{n(n+1)(2n+1)}{6}\,. \qquad (*)$$

We prove (*) by induction on n.

Induction start. Let $n = 0$. We have $0^2 = \frac{0(0+1)(2\cdot 0+1)}{6}$, and thus (*) holds when $n = 0$.

Induction step. Let $n = m + 1$. We have

$$\begin{aligned}
0^2 + 1^2 + \cdots + n^2 = \;& 0^2 + 1^2 + \cdots + m^2 + (m+1)^2 && (\text{as } m+1 = n)\\
=\; & \tfrac{m(m+1)(2m+1)}{6} + (m+1)^2 && (\text{ind. hyp.})\\
=\; & \tfrac{m(m+1)(2m+1)+6(m+1)^2}{6}\\
=\; & \tfrac{(m+1)(m(2m+1)+6(m+1))}{6}\\
=\; & \tfrac{(m+1)(2m^2+7m+6)}{6}\\
=\; & \tfrac{(m+1)(m+2)(2(m+1)+1)}{6}\\
=\; & \tfrac{(n)(n+1)(2n+1)}{6} && (\text{as } m+1 = n)
\end{aligned}$$

Thus, (*) holds when $n = m + 1$.

Section 1.4.1, Exercise 3, page 17:
Let n be the number of elements in the finite set A.

$$A \text{ has exactly } 2^n \text{ subsets.} \qquad (*)$$

We will prove (*) by induction on n.

Induction start. Let $n = 0$. Now, $\emptyset$ is the only set with 0 elements, and thus $A = \emptyset$. Furthermore, $\emptyset$ is the only subset of $\emptyset$. Thus, A has exactly one subset. We have $2^n = 2^0 = 1$. Hence, (*) holds when $n = 0$.

Induction step. Let $n = m + 1$. Fix $a \in A$ and let $A' = A \setminus \{a\}$. Now, A' has m elements. Let $B_1, \ldots, B_k$ be the subsets of A'. By our induction hypothesis, we have $k = 2^m$. For $i = 1, \ldots, k$, let $B'_i = B_i \cup \{a\}$. Obviously every set in the list $B_1, \ldots, B_k, B'_1, \ldots, B'_k$ is a subset of A. Moreover, you will find all the subsets of A in this list. Thus, the number of subsets of A is $k + k = 2^m + 2^m = 2^{m+1} = 2^n$. This proves (*) when $n = m + 1$.

Section 1.4.1, Exercise 4, page 17:

Let 0 be a constant symbol. Let f and g be function symbols of, respectively, arity 2 and arity 4. Let $\mathcal{L}$ be the language $\{0, f, g\}$.

Let t be an $\mathcal{L}$-term. We will prove that

$$t \text{ has an odd number of symbols}$$

by induction on the complexity of t.

Case: $t :\equiv 0$. Then, t has 1 symbol, and 1 is an odd number.

Case: t *is a variable.* Then, t has 1 symbol, and 1 is an odd number.

Case: $t :\equiv ft_1t_2$. Let n_1 be the number of symbols in t_1, and let n_2 be the number of symbols in t_2. The induction hypothesis states that n_1 and n_2 are odd numbers. Let $n = n_1 + n_2 + 1$. Now, n is the number of symbols in t, and moreover, n is an odd number since n_1 and n_2 are odd numbers.

Case: $t :\equiv gt_1t_2t_3t_4$. This case is similar to the preceding case. Let n_1, n_2, n_3, and n_4 be the number of symbols in respectively t_1, t_2, t_3, and t_4. We know by our induction hypothesis that n_1, n_2, n_3, and n_4 are odd numbers, and then, the number of symbols in t, that is, the number $n_1 + n_2 + n_3 + n_4 + 1$ is also odd.

Section 1.4.1, Exercise 5, page 17:

Let $\mathcal{L}$ be the language $\{0, <\}$ where 0 is a constant symbol and $<$ is a binary relation symbol. Let ϕ be an $\mathcal{L}$-formula.

$$\text{The number of symbols in } \phi \text{ is divisible by 3.} \qquad (*)$$

We prove (*) by induction on the complexity of ϕ.

Case: $\phi :\equiv =t_1t_2$ *where* t_1 *and* t_2 *are* $\mathcal{L}$*-terms.* Any $\mathcal{L}$-term has exactly one symbol (as $\mathcal{L}$ does not contain any function symbols). Thus, there are 3 symbols in the term $=t_1t_2$, and 3 is divisible by 3. We conclude that (*) holds when ϕ is in the form $=t_1t_2$.

Case: $\phi :\equiv <t_1t_2$ *where* t_1 *and* t_1 *are* $\mathcal{L}$*-terms.* This case is similar to the preceding case.

Case: $\phi :\equiv (\neg\alpha)$. Let n be the number of symbols in the formula α. Our induction hypothesis asserts that n is divisible by 3. The

number of symbols in $(\neg\alpha)$ is $n+3$, and $n+3$ is divisible by 3 when n is divisible by 3. Thus (*) holds.

Case: $\phi :\equiv (\alpha \vee \beta)$. Let n_1 be the number of symbols in the formula α, and let n_2 be the number of symbols in the formula β. By the induction hypothesis, n_1 and n_2 are divisible by 3. The number of symbols in $(\alpha \vee \beta)$ is $n_1 + n_2 + 3$, and this number is divisible by 3 as n_1 and n_2 are divisible by 3. Thus (*) holds.

Case: $\phi :\equiv (\forall x)(\alpha)$. Let n be the number of symbols in the formula α. By the induction hypothesis, n is divisible by 3. The number of symbols in $(\forall x)(\alpha)$ is $n+6$, and $n+6$ is divisible by 3 when n is divisible by 3. Thus (*) holds.

Section 1.6.1

Section 1.6.1, Exercise 2, page 26:

Let 0 be a constant symbol, let $+$ be a binary function symbol, let $<$ be a binary relation symbol, and let $\mathcal{L}$ be the language $\{0, +, <\}$.

A few words on terminology: Let $\mathfrak{A}$ be an $\mathcal{L}$-structure and let a and b be two elements in the universe of A. We will say that *a is less than b in $\mathfrak{A}$* if the relation $a <^{\mathfrak{A}} b$ holds; we will say that $a +^{\mathfrak{A}} b$ is the *value of $a + b$ in $\mathfrak{A}$*; we will also say that $a +^{\mathfrak{A}} b$ is the *sum of $a + b$ in $\mathfrak{A}$*.

We will give three different $\mathcal{L}$-structures $\mathfrak{A}$, $\mathfrak{B}$, and $\mathfrak{C}$. In all three structures, the universe will be the natural numbers together with Ingrid and Bogie, that is, the set $\mathbb{N} \cup \{\text{Ingrid}, \text{Bogie}\}$.

We define the structure $\mathfrak{A}$ by $A = \mathbb{N} \cup \{\text{Ingrid}, \text{Bogie}\}$ and

- $0^{\mathfrak{A}} = \text{Bogie}$
- $<^{\mathfrak{A}}$ is the equality relation, that is, $<^{\mathfrak{A}} = \{(x, x) \mid x \in A\}$
- $a +^{\mathfrak{A}} b = 0$ for any $a, b \in A$.

In this structure the value of $5 +^{\mathfrak{A}} \text{Ingrid}$ is the natural number 0, indeed, the sum of any two elements of the universe is 0. It is not true that Bogie is less than 0 in $\mathfrak{A}$, indeed, no element is less than another element in $\mathfrak{A}$. We should be careful before we answer the question: Is Bogie < 0? The question is ambiguous. Is Bogie less than 0 in $\mathfrak{A}$? The answer depends on how we read the symbol 0. If this symbol denotes the natural number zero, the answer is No. ($<^{\mathfrak{A}}$ is the equality relation, and 0 and Bogie are different elements of the universe, and thus, the relation Bogie $<^{\mathfrak{A}} 0^{\mathfrak{A}}$ does not hold.) If the symbol denotes the constant symbol of $\mathcal{L}$, then the answer is Yes. (The relation Bogie $<^{\mathfrak{A}} 0^{\mathfrak{A}}$ holds.) Ambiguities occur frequently in mathematical writings. Normally, the context will help you dissolve the ambiguities, but not always.

Mathematics is an exact science – it is even regarded as a science that is more exact than any other exact science. (If there are any other exact sciences? Some would say there are only *two* sciences that should count as exact sciences: mathematics and hindsight.) Mathematical statements are definitely supposed to be precise statements. So why do mathematicians not avoid ambiguous notation? Well, it is very often convenient to state a precise statement imprecisely: This may save an author a lot of time, words, and symbols; this may spare a reader for a some boring stuff that he probably does not need to read anyway, because he knows very well what the author intends to say. This is why ambiguities occur frequently in mathematical writings.

We will now define the structure $\mathfrak{B}$. The universe of $\mathfrak{B}$ is $\mathbb{N} \cup \{\text{Ingrid}, \text{Bogie}\}$. The interpretation of the constant symbol 0 is given by $0^{\mathfrak{B}} = 0$. The interpretation of the relation symbol $<$ is given by

$$<^{\mathfrak{B}} = \{(x, y) \mid x, y \in \mathbb{N} \text{ and } x < y\} \, . \qquad (*)$$

Now, should we, or should we not, tell the reader that one occurrence of the symbol $<$ in (*) denotes standard strict ordering of the natural numbers? whereas the other occurrence denotes the relation symbol in $\mathcal{L}$? The notation used in (*) is ambiguous and that might cause some confusion. On the other hand, there is only one way of reading of (*) that makes sense. So maybe there is no need make any fuss about this ... and write some boring stuff that the reader probably will skip anyway ... To complete the definition of $\mathfrak{B}$, we must give the interpretation $+^{\mathfrak{B}}$ of the function symbol $+$. Recall that $+^{\mathfrak{B}}$ has to be a total function, and we cannot, e.g., say that $+^{\mathfrak{B}}$ is standard addition on natural numbers. If we do so, $+^{\mathfrak{B}}$ will not be a total function as values like $5 + \text{Ingrid}$ and $\text{Ingrid} + \text{Bogie}$ will be undefined. But we may say that

$$a +^{\mathfrak{B}} b = \begin{cases} a + b & \text{if } a, b \in \mathbb{N} \text{ (standard addition)} \\ 17 & \text{otherwise.} \end{cases}$$

Now we have a structure $\mathfrak{B}$ where the value of $5 + \text{Ingrid}$ is 17. What will be a reasonable answer to the potential ambiguous question: Is $\text{Bogie} < 0$? Note that $0^{\mathfrak{B}} = 0$. Thus, we do not run into the same problems as we did when we tried to answer this question with respect to the structure $\mathfrak{A}$. Now we work in a context where this question only has one correct answer. The answer is Yes if the pair $(\text{Bogie}, 0)$ is in the set $<^{\mathfrak{B}}$. The answer is No if the pair is not in set. Thus, we say No, Bogie is not less than 0 in the structure $\mathfrak{B}$.

We define $\mathcal{L}$-structure $\mathfrak{C}$ by $C = \mathbb{N} \cup \{\text{Ingrid}, \text{Bogie}\}$ and

- $0^{\mathfrak{C}} = 0$

- $$<^{\mathfrak{C}} \;=\; \{(x,y) \mid x,y \in \mathbb{N} \text{ and } x > y\} \;\cup\; \{(x,y) \mid x \in \{\text{Ingrid}, \text{Bogie}\} \text{ and } y \in \mathbb{N}\} \;\cup\; \{(\text{Bogie}, \text{Ingrid})\}$$

- $$a +^{\mathfrak{C}} b = \begin{cases} b & \text{if } a <^{\mathfrak{C}} b \\ a & \text{otherwise.} \end{cases}$$

In this structure we find exactly one element that is less than Ingrid (Bogie), and we find infinitely many elements that are not less than Ingrid (every element except Bogie) We find exactly four elements that are not less than 3 (0, 1, 2 and 3), and we find infinitely many elements that are less than 3 (every except 0, 1, 2 and 3). 0 is not less than any other element in the universe, We have

$$\text{Bogie} \;<^{\mathfrak{C}}\; \text{Ingrid} \;<^{\mathfrak{C}}\; \ldots \;<^{\mathfrak{C}}\; 3 \;<^{\mathfrak{C}}\; 2 \;<^{\mathfrak{C}}\; 1 \;<^{\mathfrak{C}}\; 0\,.$$

The value of $5 + \text{Ingrid}$ is 5.

Section 1.6.1, Exercise 5, page 27:

(a) Two $\mathcal{L}$-structures $\mathfrak{A}$ and $\mathfrak{B}$ are *isomorphic*, written $\mathfrak{A} \cong \mathfrak{B}$, if there exists a bijection $\imath : A \to B$ such that

(A) $\imath(c^{\mathfrak{A}}) = c^{\mathfrak{B}}$ for any constant symbol c of $\mathcal{L}$

(B) $\imath(f^{\mathfrak{A}}(a_1, \ldots, a_n)) = f^{\mathfrak{B}}(\imath(a_1), \ldots, \imath(a_n))$ for any n-ary function symbol f of $\mathcal{L}$ and any $a_1, \ldots, a_n \in A$

(C) $\langle a_1, \ldots, a_n \rangle \in R^{\mathfrak{A}} \Leftrightarrow \langle \imath(a_1), \ldots, \imath(a_n) \rangle \in R^{\mathfrak{B}}$ for any n-ary relation symbol R of $\mathcal{L}$ and any $a_1, \ldots, a_n \in A$.

Now we have defined the relation $\cong$. Next, we will prove that $\cong$ is an equivalence relation. We have to prove that

(1) $\mathfrak{A} \cong \mathfrak{A}$ (the relation is reflexive)

(2) if $\mathfrak{A} \cong \mathfrak{B}$ and $\mathfrak{B} \cong \mathfrak{C}$, then $\mathfrak{A} \cong \mathfrak{C}$ (the relation is transitive)

(3) if $\mathfrak{A} \cong \mathfrak{B}$, then $\mathfrak{B} \cong \mathfrak{A}$ (the relation is symmetric)

Any relation that is reflexive, transitive, and symmetric, is by definition an equivalence relation.

First, we prove that $\cong$ is a reflexive relation. Then we have to come up with a bijection $\imath : A \to A$ which satisfies requirements (A), (B), and (C). This is easy. Let $\imath(a) = a$ for any $a \in A$. It is obvious that $\imath$ is a bijection, and it is easy to see that $\imath$ satisfies the three requirements. For example, to check that (B) is satisfied, we have to check that

$$\imath(f^{\mathfrak{A}}(a_1, \ldots, a_n)) = f^{\mathfrak{A}}(\imath(a_1), \ldots, \imath(a_n))$$

but this equality holds trivially when $\imath(x) = x$.

Next, we prove that $\cong$ is a symmetric relation. In order to do this, we assume that $\mathfrak{A} \cong \mathfrak{B}$, that is, we assume that we have a bijection $\imath : A \to B$ that satisfies (A), (B), and (C). We will prove that $\mathfrak{B} \cong \mathfrak{A}$ follows from this assumption. Thus, we have to find a bijection $\jmath : B \to A$ satisfying (A), (B), and (C). An obvious candidate for $\jmath$ is the inverse of $\imath$: let $\jmath(x) = \imath^{-1}(x)$. It is obvious that $\jmath$ is a bijection from B into A. We will now show carefully that $\jmath$ indeed satisfies (A), (B), and (C).

First, we will prove that $\jmath$ satisfies (A), that is, we will prove that $\jmath(c^{\mathfrak{B}}) = c^{\mathfrak{A}}$ for any constant symbol c of $\mathcal{L}$. We know that $\imath(c^{\mathfrak{A}}) = c^{\mathfrak{B}}$ since $\imath$ is an isomorphism from $\mathfrak{A}$ to $\mathfrak{B}$. Thus, we have

$$\begin{aligned} \jmath(c^{\mathfrak{B}}) &= \imath^{-1}(c^{\mathfrak{B}}) && \text{(def. of } \jmath\text{)} \\ &= \imath^{-1}(\imath(c^{\mathfrak{A}})) && \text{(since } \imath(c^{\mathfrak{A}}) = c^{\mathfrak{B}}\text{)} \\ &= c^{\mathfrak{A}} && \text{(since } \imath^{-1}(\imath(x)) = x \text{)} \end{aligned}$$

and we see that $\jmath$ satisfies (A).

We will now prove that $\jmath$ satisfies (B), that is, we will prove that

$$\jmath(f^{\mathfrak{B}}(b_1, \ldots, b_n)) = f^{\mathfrak{A}}(\jmath(b_1), \ldots, \jmath(b_n)) \qquad (*)$$

for any n-ary function symbol f of $\mathcal{L}$ and any $b_1, \ldots, b_n \in B$. To improve the readability, we will let n be 1 in our proof. It is easy to generalize the proof to an arbitrary n greater than 1.

Pick an arbitrary $b \in B$. Since $\imath$ is a bijection from A to B, there exists a unique element in A that $\imath$ maps to b. Let a denote this element. Thus, $\imath(a) = b$, and moreover, since $\imath$ is an isomorphism from A to B, we also have $\imath(f^{\mathfrak{A}}(a)) = f^{\mathfrak{B}}(\imath(a))$. Hence

$$\begin{aligned} \jmath(f^{\mathfrak{B}}(b)) &= \imath^{-1}(f^{\mathfrak{B}}(b)) && \text{(def. of } \jmath\text{)} \\ &= \imath^{-1}(f^{\mathfrak{B}}(\imath(a))) && \text{(since } \imath(a) = b\text{)} \\ &= \imath^{-1}(\imath(f^{\mathfrak{A}}(a))) && \text{(since } \imath(f^{\mathfrak{A}}(a)) = f^{\mathfrak{B}}(\imath(a)) \text{)} \\ &= f^{\mathfrak{A}}(a) && \text{(since } \imath^{-1}(\imath(x)) = x \text{)} \\ &= f^{\mathfrak{A}}(\imath^{-1}(b)) && \text{(since } \imath(a) = b\text{)} \\ &= f^{\mathfrak{A}}(\jmath(b)) && \text{(def. of } \jmath\text{)} \end{aligned}$$

Thus, (*) holds, and $\jmath$ satisfies (B).

Next, we prove that $\jmath$ satisfies (C). Then, we have to prove that

$$\langle b_1, \ldots, b_n \rangle \in R^{\mathfrak{B}} \Leftrightarrow \langle \jmath(b_1), \ldots, \jmath(b_n) \rangle \in R^{\mathfrak{A}} \qquad (**)$$

for any n-ary relation symbol R of $\mathcal{L}$ and any $b_1, \ldots, b_n \in B$. Again, we will let n be 1 in our proof. It is easy to generalize the proof to relation symbols of arity greater than 1. Recall that we know that

$$a \in R^{\mathfrak{A}} \quad \Leftrightarrow \quad \imath(a) \in R^{\mathfrak{B}}$$

as since $\imath$ is an isomorphism from A to B. Now, pick an arbitrary $b \in B$, and let a be the unique element in A such that the bijection i maps a to b. Then, we have

$$\begin{array}{lcll} b \in R^{\mathfrak{B}} & \Leftrightarrow & \imath(a) \in R^{\mathfrak{B}} & (\text{since } \imath(a) = b) \\ & \Leftrightarrow & a \in R^{\mathfrak{A}} & (\text{since } \imath \text{ is an isomorphism from } A \text{ to } B) \\ & \Leftrightarrow & \imath^{-1}(b) \in R^{\mathfrak{A}} & (\text{since } \imath(a) = b) \\ & \Leftrightarrow & \jmath(b) \in R^{\mathfrak{A}} & (\text{def. of } \jmath) \end{array}$$

This proves that (**) holds, and thus, $\jmath$ satisfies (A), (B), and (C). We conclude that $\cong$ is a symmetric relation.

We also have to prove that $\cong$ is a transitive relation. Here is how to do that: Let $\imath : A \to B$ be an isomorphism from $\mathfrak{A}$ to $\mathfrak{B}$. Let $\jmath : B \to C$ be an isomorphism from $\mathfrak{B}$ to $\mathfrak{C}$. Define $k : A \to C$ by $k(x) = \jmath(\imath(x))$, and prove that k is an isomorphism from $\mathfrak{A}$ to $\mathfrak{C}$. We skip the details.

(c) Let $\mathcal{L}$ be the language $\{+, \cdot\}$ where $+$ and $\cdot$ are binary function symbols. Let $\mathfrak{A}$ be the $\mathcal{L}$-structure where the universe is the set of natural numbers, and $+^{\mathfrak{A}}$ and $\cdot^{\mathfrak{A}}$ are, respectively, standard addition and standard multiplication on naturals. Let $\mathfrak{B}$ be the $\mathcal{L}$-structure where the universe is the set of real numbers, and $+^{\mathfrak{B}}$ and $\cdot^{\mathfrak{B}}$ are, respectively, standard addition and standard multiplication on reals.

Then, $\mathfrak{A}$ is not isomorphic to $\mathfrak{B}$. Why? Because there does not exist a bijection between the natural numbers and the real numbers.

(d) Let $\mathcal{L}$ be the language $\{+, \cdot\}$ where $+$ and $\cdot$ are binary function symbols. Let $\mathfrak{A}$ be the $\mathcal{L}$-structure where the universe is the set $\mathbb{N}$ of natural numbers, and $+^{\mathfrak{A}}$ and $\cdot^{\mathfrak{A}}$ are, respectively, standard addition and standard multiplication on $\mathbb{N}$. Let $\mathfrak{B}$ be the $\mathcal{L}$-structure where the universe is the set $\mathbb{N}$ of natural numbers, and

- $x +^{\mathfrak{B}} y = \max(x, y)$ for any $x, y \in \mathbb{N}$
- $x \cdot^{\mathfrak{B}} y = \min(x, y)$ for any $x, y \in \mathbb{N}$.

Now, $\mathfrak{A}$ is not isomorphic to $\mathfrak{B}$. To show this, assume for the sake of contradiction that $\imath : A \to B$ is an isomorphism from $\mathfrak{A}$ to $\mathfrak{B}$. Then, we have

$$\imath(2) \;=\; \imath(1 +^{\mathfrak{A}} 1) \;=\; \imath(1) +^{\mathfrak{B}} \imath(1) \;=\; \imath(1)\,.$$

The first equality holds since $+^{\mathfrak{A}}$ is standard addition; the second equality holds since $\imath$ is an isomorphism; the third equality holds since $x +^{\mathfrak{B}} x = x$ for any $x \in \mathbb{N}$. We see that $\imath(2) = \imath(1)$, and this contradicts that $\imath$ is a bijection.

Section 1.7.1

Section 1.7.1, Exercise 7, page 33:

$$\begin{array}{rl}
\mathfrak{A} \models (\exists x)(\alpha)[s] & \\
\Updownarrow & (\text{since } (\exists x)(\alpha) :\equiv (\neg(\forall x)(\neg\alpha))) \\
\mathfrak{A} \models (\neg(\forall x)(\neg\alpha))[s] & \\
\Updownarrow & (\text{Def. 1.7.4}) \\
\mathfrak{A} \not\models (\forall x)(\neg\alpha)[s] & \\
\Updownarrow & (\text{Def. 1.7.4}) \\
\mathfrak{A} \models (\neg\alpha)[s[x|b]] \text{ does not hold for every } b \in A & \\
\Updownarrow & (\text{obvious}) \\
\text{there exists } a \in A \text{ such that } \mathfrak{A} \not\models (\neg\alpha)[s[x|a]] & \\
\Updownarrow & (\text{Def. 1.7.4}) \\
\text{there exists } a \in A \text{ such that } \mathfrak{A} \models \alpha[s[x|a]] &
\end{array}$$

Section 1.8.1

Section 1.8.1, Exercise 4, page 36:

We will show

$$x \text{ is substitutable for } x \text{ in } \phi \qquad (*)$$

by induction over the complexity of the formula ϕ.

Case: ϕ is atomic. Then (*) holds by clause 1 of Definition 1.8.3.

Case: $\phi :\equiv (\neg\alpha)$. Now, x is substitutable for x in α by our induction hypothesis, and then (*) holds by clause 2 of Definition 1.8.3.

Case: $\phi :\equiv (\alpha \vee \beta)$. By the induction hypothesis we know that

- x is substitutable for x in α
- x is substitutable for x in β

and then (*) holds by clause 3 of Definition 1.8.3.

Case: $\phi :\equiv (\forall y)(\alpha)$. We split the proof into two subcases:

(i) x and y are the same variable (e.g., v_{17})

(ii) x and y are different variables.

In case (i), x is not free in ϕ, and then (*) holds by (a) of Clause 4 of Definition 1.8.3. (We do not need the induction hypothesis in this case.) Let us consider case (ii). In this case y does not occur in the term x, and by our induction hypothesis we know that x is substitutable for x in α. Thus, (*) holds by (b) of Clause 4 of Definition 1.8.3.

Section 1.8.1, Exercise 6, page 36:

Let R be a binary relation symbol. Now, $((\forall y)(Rxy))^x_y$ is $(\forall y)(Ryy)$, and $((\forall y)(Ryy))^y_x$ is $(\forall y)(Ryy)$. Hence, $(((\forall y)(Rxy))^x_y)^y_x$ is $(\forall y)(Ryy)$. It turns out that $(\phi^x_y)^y_x$ is ϕ if, and only if, y is substitutable for x in ϕ.

Section 1.9.1

Section 1.9.1, Exercise 1, page 38:

We will show that we have $\{\alpha, \alpha \to \beta\} \models \beta$ for any formulas α and β. By Definition 1.9.1, we have to prove that

$$\mathfrak{A} \models \{\alpha, \alpha \to \beta\} \;\; \Rightarrow \;\; \mathfrak{A} \models \beta$$

holds for any structure $\mathfrak{A}$.

We have

$$\begin{array}{rl}
\mathfrak{A} \models \{\alpha, \alpha \to \beta\} & \\
\Updownarrow & \text{(def. of } \to\text{)} \\
\mathfrak{A} \models \{\alpha, \neg\alpha \vee \beta\} & \\
\Updownarrow & \text{(Def. 1.7.9)} \\
\mathfrak{A} \models \{\alpha, \neg\alpha \vee \beta\}[s] \text{ for every } s & \\
\Updownarrow & \text{(Def. 1.7.9)} \\
\mathfrak{A} \models \alpha[s] \text{ and } \mathfrak{A} \models \neg\alpha \vee \beta[s] \text{ for every } s & \\
\Updownarrow & \text{(Def. 1.7.4)} \\
\mathfrak{A} \models \alpha[s] \text{ and } \mathfrak{A} \models \beta[s] \text{ for every } s & \\
\Downarrow & \\
\mathfrak{A} \models \beta[s] \text{ for every } s & \\
\Updownarrow & \text{(Def. 1.7.9)} \\
\mathfrak{A} \models \beta &
\end{array}$$

Section 1.9.1, Exercise 2, page 38:

We prove that the formula $x = x$ is valid.

We have to prove that $\mathfrak{A} \models x = x[s]$ holds for any structure $\mathfrak{A}$ and any assignment function s. (See Definition 1.9.2.) So, let $\mathfrak{A}$ be an arbitrary structure, and let s be an arbitrary assignment function. Definition 1.7.4 states: for any terms t_1 and t_2, we have $\mathfrak{A} \models t_1 = t_2[s]$ if $\overline{s}(t_1)$ is the same element of the universe A as $\overline{s}(t_2)$. Now, $\overline{s}(x)$ is of course the same element as $\overline{s}(x)$, and thus, $\mathfrak{A} \models x = x[s]$ holds.

Next, we prove that the formula $x = y$ is not valid. Let $\mathfrak{A}$ be a structure where the universe is the set of natural numbers. Let s be the assignment function such that $s(x) = 17$ and $s(y) = 314$. By Definition 1.7.4, we have $\mathfrak{A} \not\models x = y[s]$. Thus, it is not the case that $\mathfrak{A} \models x = y[s]$ holds for any structure $\mathfrak{A}$ and any assignment function s. Then, by Definition 1.9.2, $x = y$ is not a valid formula.

Finally, we prove that the formula $\neg x = y$ is not valid. Let $\mathfrak{A}$ be a structure where the universe is the natural numbers. Let s be the assignment function such that $s(x) = 17$ and $s(y) = 17$. Then, by Definition 1.7.4, we have $\mathfrak{A} \not\models \neg x = y[s]$. Thus, it is not the case that $\mathfrak{A} \models \neg x = y[s]$ holds for any structure $\mathfrak{A}$ and any assignment function s. Then, by Definition 1.9.2, $\neg x = y$ is not a valid formula.

Section 1.9.1, Exercise 4, page 38:

(a) Assume $\models \phi \to \psi$. (We will show that $\phi \models \psi$.) Then, by Definition 1.9.2, for every $\mathfrak{A}$ and every s, we have

$$\mathfrak{A} \models \phi \to \psi\,[s]\ .$$

Then, for every $\mathfrak{A}$ and every s, we have

$$\mathfrak{A} \models \phi[s] \quad \Rightarrow \quad \mathfrak{A} \models \psi[s]\ .$$

Then, for every $\mathfrak{A}$, we have

$$\mathfrak{A} \models \phi[s_1] \text{ for every } s_1 \quad \Rightarrow \quad \mathfrak{A} \models \psi[s_2] \text{ for every } s_2\ .$$

Then, by Definition 1.7.9, for every $\mathfrak{A}$, we have

$$\mathfrak{A} \models \phi \quad \Rightarrow \quad \mathfrak{A} \models \psi\ .$$

Then, by Definition 1.9.1, we have $\phi \models \psi$.

(b) Let x, y, z, w be variables. We will prove that

$$x < y \models z < w \quad \text{and} \quad \not\models x < y \to z < w$$

First we prove that

$$\mathfrak{A} \models x < y \quad \Rightarrow \quad \mathfrak{A} \models z < w \qquad (*)$$

holds for every $\mathfrak{A}$.

Assume $\mathfrak{A} \models x < y$. By Definition 1.7.1, we have

$$\mathfrak{A} \models x < y[s] \text{ for every } s.$$

Thus, $<^{\mathfrak{A}} = A \times A$, that is, the relation $a <^{\mathfrak{A}} b$ holds for any a, b in the universe of $\mathfrak{A}$. Thus

$$\mathfrak{A} \models z < w[s] \text{ for every } s.$$

Thus, by Definition 1.7.9, we have $\mathfrak{A} \models z < w$. This shows that (*) holds.

It follows from (*) and Definition 1.9.1 that $x < y \models z < w$.

Next, we will prove that $\not\models x < y \to z < w$. Let $\mathfrak{N}$ be the model where the universe is $\mathbb{N}$ and $<^{\mathfrak{N}}$ is the standard ordering of $\mathbb{N}$. Let s be an assignment function such that $s(x) = s(w) = 0$ and $s(y) = s(z) = 1$. Then we have $\mathfrak{N} \not\models x < y \to z < w[s]$. Furthermore, we have $\mathfrak{N} \not\models x < y \to z < w$ by Definition 1.7.9. Finally, we have $\not\models x < y \to z < w$ by Definition 1.9.2.

Section 2.4.3

Section 2.4.3, Exercise 4, page 54:

Note that

$$\phi_t^x \to (\neg(\forall x)((\neg\phi)))$$

can derived from the premise $(\forall x)((\neg\phi)) \to \neg\phi_t^x$ by the rule (PC). Moreover, note that $(\exists x)(\phi)$ is an abbreviation for $(\neg(\forall x)((\neg\phi)))$. Hence, (Q2) can be derived from (Q1) by a one application of (PC).

Section 2.4.3, Exercise 6, page 54:

Let $\mathcal{L}$ be a first-order order language, let ϕ be an $\mathcal{L}$-formula, and let $A_1, \ldots, A_n$ be the propositional variables that occur in the formula ϕ_P.

Now, each variable in the list $A_1, \ldots, A_n$ corresponds to a subformula of ϕ. For $i = 1, \ldots, n$, let ψ_i denote the subformula that corresponds to A_i. For any $\mathcal{L}$-structure $\mathfrak{A}$, we define the valuation $v_{\mathfrak{A}}$ by

$$v_{\mathfrak{A}}(A_i) = \begin{cases} T & \text{if } \mathfrak{A} \models \psi_i \\ F & \text{otherwise.} \end{cases}$$

(Claim)

$$\mathfrak{A} \not\models \phi \quad \Leftrightarrow \quad \bar{v}_{\mathfrak{A}}(\phi_P) = F\ .$$

Note that this claim is equivalent to

$$\mathfrak{A} \models \phi \quad \Leftrightarrow \quad \bar{v}_{\mathfrak{A}}(\phi_P) = T \ .$$

We will now prove the claim by induction over the complexity of ϕ.

Case: ϕ is an atomic formula. The claim follows straightforwardly from the definition of $v_{\mathfrak{A}}$.

Case: ϕ is of the form $(\forall x)(\alpha)$. The claim follows straightforwardly from the definition of $v_{\mathfrak{A}}$.

Case: ϕ is of the form $(\neg\alpha)$. We have

$$\mathfrak{A} \not\models (\neg\alpha) \quad \Leftrightarrow \quad \mathfrak{A} \models \alpha \quad \Leftrightarrow \quad \bar{v}_{\mathfrak{A}}(\alpha_P) = T \quad \Leftrightarrow \quad \bar{v}_{\mathfrak{A}}((\neg\alpha)_P) = F \ .$$

The second equivalence follows by our induction hypothesis.

Case: ϕ is of the form $(\alpha \vee \beta)$.

$$\begin{aligned} \mathfrak{A} \not\models (\alpha \vee \beta) \quad &\Leftrightarrow \quad \mathfrak{A} \not\models \alpha \text{ and } \mathfrak{A} \not\models \beta \\ &\Leftrightarrow \quad \bar{v}_{\mathfrak{A}}(\alpha_P) = F \text{ and } \bar{v}_{\mathfrak{A}}(\beta_P) = F \qquad \text{(ind. hyp.)} \\ &\Leftrightarrow \quad \bar{v}_{\mathfrak{A}}((\alpha \vee \beta)_P) = F \ . \end{aligned}$$

This completes the proof of (Claim).

Now, assume that the first-order formula θ is not valid. By the definition of validity, there exists a structure $\mathfrak{A}$ such that $\mathfrak{A} \not\models \theta$. By (Claim) there exists a valuation v such that $\bar{v}(\theta_P) = F$. Thus, by the definition of a tautology, θ_P is not a tautology. Hence, if θ_P is a tautology, then θ is valid.

Section 2.7.1

Section 2.7.1, Exercise 4, page 66:

Let $\bot \equiv \forall x[x = x] \wedge \neg\forall x[x = x]$, and let η be a sentence. We prove that

$$\Sigma \vdash \eta \quad \Leftrightarrow \quad \Sigma \cup \{\neg\eta\} \vdash \bot \ .$$

(Does this equivalence hold if η is not a sentence and may contain free variables?)

Assume $\Sigma \vdash \eta$. Then we have $\Sigma \cup \{\neg\eta\} \vdash \eta$. Obviously we also have $\Sigma \cup \{\neg\eta\} \vdash \neg\eta$. Now, $\bot$ follows tautologically from η and $\neg\eta$. (Any formula follows tautologically from η and $\neg\eta$.) Thus, since (PC) is an inference rule in our deductive system, we have $\Sigma \cup \{\neg\eta\} \vdash \bot$.

Assume $\Sigma \cup \{\neg\eta\} \vdash \bot$. By the Deduction Theorem, we have $\Sigma \vdash \neg\eta \to \bot$. Now, η follows tautologically from $\neg\eta \to \bot$. Thus, as (PC) is a rule of inference, we have $\Sigma \vdash \eta$.

Section 2.7.1, Exercise 5, page 66:

We will be generous and give a number of different solutions of this exercise.

Solution I.

We use Exercise 4. Let $\Sigma = \emptyset$, and let $\eta :\equiv [(\forall x)P(x)] \to [(\exists x)P(x)]$. Here is a deduction of $\bot$ from $\neg\eta$:

1.	$\neg([(\forall x)P(x)] \to [(\exists x)P(x)])$	
2.	$(\forall x)P(x)$	1, (PC)
3.	$\neg(\exists x)P(x)$	1, (PC)
4.	$\neg\neg(\forall x)\neg P(x)$	3, $:\equiv$
5.	$(\forall x)\neg P(x)$	4, (PC)
6.	$[(\forall x)\neg P(x)] \to \neg P(x)$	(Q1)
7.	$\neg P(x)$	5, 6, (PC)
8.	$[(\forall x)P(x)] \to P(x)$	(Q1)
9.	$P(x)$	2, 8, (PC)
10.	$\bot$	7, 9, (PC)

This shows that $\neg([(\forall x)P(x)] \to [(\exists x)P(x)]) \vdash \bot$. We know from Exercise 4 that

$$\Sigma \vdash \eta \quad \Leftrightarrow \quad \Sigma \cup \{\neg\eta\} \vdash \bot\ .$$

Thus we have $\vdash [(\forall x)P(x)] \to [(\exists x)P(x)]$.

Solution II.

We use the Deduction Theorem. This is a deduction of $(\exists x)P(x)$ from $\{(\forall x)P(x)\}$:

1.	$(\forall x)P(x)$	
2.	$[(\forall x)P(x)] \to P(x)$	(Q1)
3.	$P(x)$	1, 2, (PC)
4.	$P(x) \to [(\exists x)P(x)]$	Q2
5.	$(\exists x)P(x)$	3, 4, (PC)

This proves that $\{(\forall x)P(x)\} \vdash (\exists x)P(x)$. By the Deduction Theorem, we have $\vdash [(\forall x)P(x)] \to [(\exists x)P(x)]$.

Solution III.

We have a deduction of $\forall x[P(x)] \to \exists x[P(x)]$ from $\emptyset$:

1.	$[(\forall x)P(x)] \to P(x)$	(Q1)
2.	$P(x) \to [(\exists x)P(x)]$	(Q2)
3.	$[(\forall x)P(x)] \to [(\exists x)P(x)]$	1, 2, (PC)

Solution IV.

We have a derivation of $\forall x[Px] \to \exists x[Px]$ from $\emptyset$:

1.	$[(\forall x)P(x)] \to P(x)$	(Q1)
2.	$[(\forall x)\neg P(x)] \ \to \ \neg P(x)$	(Q1)
3.	$[(\forall x)P(x)] \to [\neg(\forall x)\neg P(x)]$	1, 2, (PC)
4.	$[(\forall x)P(x)] \to [(\exists x)P(x)]$	3, $:\equiv$

The reader should study how Solution III relates to Solution IV. See also Exercise 4 in Section 2.4.3.

Section 2.7.1, Exercise 6, page 66:
This is a deduction of $(\forall y)(\forall z)P(z,y)$ from $\{(\forall x)(\forall y)P(x,y)\}$:

1.	$(\forall x)(\forall y)P(x,y)$	
2.	$[(\forall x)(\forall y)P(x,y)] \to (\forall y)P(z,y)$	(Q1)
3.	$[(\forall y)P(z,y)] \to P(z,y)$	(Q1)
4.	$P(z,y)$	1, 2, 3, (PC)
5.	$(x = x \vee \neg x = x) \to P(z,y)$	4, (PC)
6.	$(x = x \vee \neg x = x) \to [(\forall z)P(z,y)]$	5, (QR)
7.	$(x = x \vee \neg x = x) \to [(\forall y)(\forall z)P(z,y)]$	6, (QR)
8.	$(\forall y)(\forall z)P(z,y)$	7, (PC)

This shows that $(\forall x)(\forall y)P(x,y) \vdash (\forall y)(\forall z)P(z,y)$.

Section 2.7.1, Exercise 7, page 66:

Solution I.

We have a derivation from $\{(\forall x)(P(x)) \wedge (\forall x)(Q(x))\}$ of $(\forall x)[P(x) \wedge Q(x)]$:

1.	$(\forall x)(P(x)) \wedge (\forall x)(Q(x))$	
2.	$(\forall x)P(x)$	1, (PC)
3.	$[(\forall x)P(x)] \rightarrow P(x)$	(Q1)
4.	$P(x)$	2, 3, (PC)
5.	$(\forall x)Q(x)$	1, (PC)
6.	$[(\forall x)Q(x)] \rightarrow Q(x)$	(Q1)
7.	$Q(x)$	5, 6, (PC)
8.	$P(x) \wedge Q(x)$	4, 7, (PC)
9.	$[y = y \vee \neg y = y] \rightarrow [P(x) \wedge Q(x)]$	8 (PC)
10.	$[y = y \vee \neg y = y] \ \rightarrow \ (\forall x)[P(x) \wedge Q(x)]$	9, QR
11.	$(\forall x)[P(x) \wedge Q(x)]$	10, (PC)

This shows that

$$\{(\forall x)(P(x)) \wedge (\forall x)(Q(x))\} \vdash (\forall x)[P(x) \wedge Q(x)] \, .$$

By the Deduction Theorem, we have

$$\vdash (\forall x)(P(x)) \wedge (\forall x)(Q(x)) \rightarrow (\forall x)[P(x) \wedge Q(x)] \, .$$

Solution II.

1.	$(\forall x)(P(x)) \rightarrow P(x)$	(Q1)
2.	$(\forall x)(Q(x)) \rightarrow Q(x)$	(Q1)
3.	$(\forall x)(P(x)) \wedge (\forall x)(Q(x)) \ \rightarrow \ P(x) \wedge Q(x)$	1, 2, (PC)
4.	$(\forall x)(P(x)) \wedge (\forall x)(Q(x)) \ \rightarrow \ (\forall x)[P(x) \wedge Q(x)]$	3, (QR)

Section 2.8.1

Section 2.8.1, Exercise 4, page 70:

The Axiom of Extensionality:

$$(\forall a)(\forall b)[\,(\forall x)[x \in a \ \leftrightarrow \ x \in b] \ \rightarrow \ a = b\,]$$

Note that the converse implication, that is, the formula

$$(\forall a)(\forall b)[\, a = b \;\; \rightarrow \;\; (\forall x)[x \in a \;\leftrightarrow\; x \in b]\,]$$

holds in any structure.

The Null Set Axiom:

$$(\exists a)(\forall x)\neg x \in a$$

The Pair Set Axiom:

$$(\forall a)(\forall b)(\exists c)\,[\, a \in c \;\; \wedge \;\; b \in c \;\; \wedge \;\; (\forall x)[x \in c \;\rightarrow\; (x = a \vee x = b)]\,]$$

The Axiom of Union:

$$(\forall a)(\exists b)(\forall x)[\, x \in b \;\; \leftrightarrow \;\; (\exists y)[y \in a \;\wedge\; x \in y]\,]$$

Let

$$a \subseteq b \;:\equiv\; (\forall x)[x \in a \;\rightarrow\; x \in b]$$

The Power Set Axiom:

$$(\forall a)(\exists b)(\forall x)[\, x \subseteq a \;\; \rightarrow \;\; x \in b\,]$$

The version of the Power Set Axiom above states that for any set a there exists a set b that contains all subset of a. Other axioms of set theory will assure that there exists a set that contains all subset of a and nothing but the subsets of a. If we want the Power Set Axiom to state that for any set a there exists a set b that contains *exactly the subsets* of a, we should use a bi-implication:

$$(\forall a)(\exists b)(\forall x)[\, x \subseteq a \;\; \leftrightarrow \;\; x \in b\,]$$

Section 2.8.1, Exercise 5, page 71:

The proof of Lemma 2.8.4(4). We will prove that, for any $a, b \in \mathbb{N}$, we have

$$a \not< b \quad \Rightarrow \quad N \vdash \neg\overline{a} < \overline{b}\,. \qquad (*)$$

We will do induction on b. When $b = 0$, we have $a \not< b$, but we also have $N \vdash \neg\overline{a} < \overline{b}$ as $(\forall x)\neg x < 0$ is an axiom of N.

Now, assume that $b = c + 1$. Our induction hypothesis will be

$$a \not< c \quad \Rightarrow \quad N \vdash \neg\overline{a} < \overline{c}\,. \qquad \text{(ind. hyp)}$$

Our goal is to show (*). Thus, we assume $a \not< b$ (and argue that $N \vdash \neg\overline{a} < \overline{b}$). When $a \not< b$, we also have $a \not< c = b - 1$ and $a \neq c = b - 1$. By (ind. hyp) and Lemma 2.8.4(2), we have

$$N \vdash \neg a < c \quad \text{and} \quad N \vdash a \neq c\,. \tag{i}$$

By Axiom N10, we have

$$N \vdash \overline{a} < \overline{b} \quad \leftrightarrow \quad \overline{a} < \overline{c} \ \vee \ \overline{a} = \overline{c} \tag{ii}$$

(Recall that $b = c + 1$. Thus, (ii) is what we get when we instantiate N10 with $\overline{a}$ for x and $\overline{c}$ for y.) By (i), (ii) and (PC), we have

$$N \vdash \neg\overline{a} < \overline{b}\,.$$

The completes the proof of Lemma 2.8.4(4).

The proof of Lemma 2.8.4(6). We will prove that, for any $a, b \in \mathbb{N}$, we have $N \vdash \overline{a} \cdot \overline{b} = \overline{a \cdot b}$.

We will do induction on b. When $b = 0$, we have $N \vdash \overline{a} \cdot \overline{b} = \overline{a \cdot b}$ by Axiom N5.

Next, we assume

$$N \vdash \overline{a} \cdot \overline{b} = \overline{a \cdot b}\,. \tag{ind. hyp}$$

(We will prove $N \vdash \overline{a} \cdot \overline{b+1} = \overline{a \cdot (b+1)}$.)

By Axiom N6, we have

$$N \vdash \overline{a} \cdot \overline{b+1} = (\overline{a} \cdot \overline{b}) + \overline{a}\,. \tag{i}$$

(We get (i) when we instantiate N6 with $\overline{a}$ for x and $\overline{b}$ for y.) By (E1), (E2), (E3) and other logical axioms, we have

$$\vdash (\overline{a} \cdot \overline{b}) = \overline{a \cdot b} \ \wedge \ \overline{a} = \overline{a} \quad \rightarrow \quad (\overline{a} \cdot \overline{b}) + \overline{a} = \overline{a \cdot b} + \overline{a}\,. \tag{ii}$$

and

$$\vdash \overline{a} = \overline{a}\,. \tag{iii}$$

and

$$\vdash \overline{a} \cdot \overline{b+1} = (\overline{a} \cdot \overline{b}) + \overline{a} \ \wedge \ (\overline{a} \cdot \overline{b}) + \overline{a} = \overline{a \cdot b} + \overline{a} \quad \rightarrow \quad \overline{a} \cdot \overline{b+1} = \overline{a \cdot b} + \overline{a}\,. \tag{iv}$$

(Note that the formula in (iv) is an instantiation of $x = y \wedge y = z \ \rightarrow \ x = z$.) By (ind. hyp), (ii), (iii) and (PC) we have

$$N \vdash (\overline{a} \cdot \overline{b}) + \overline{a} = \overline{a \cdot b} + \overline{a}\,. \tag{v}$$

By (i), (iv), (v) and (PC) we have

$$N \vdash \overline{a} \cdot \overline{b+1} = \overline{a \cdot b} + \overline{a}\,. \tag{vi}$$

Now, note that $(a \cdot b) + a = a \cdot (b+1)$, and thus Lemma 2.8.4(5) yields

$$N \vdash (\overline{a \cdot b}) + \overline{a} = \overline{a \cdot (b+1)}\,. \tag{vii}$$

By the logical axioms, we have

$$\vdash \overline{a} \cdot \overline{b+1} = \overline{a \cdot b} + \overline{a} \;\wedge\; (\overline{a \cdot b}) + \overline{a} = \overline{a \cdot (b+1)} \;\rightarrow\; \overline{a} \cdot \overline{b+1} = \overline{a \cdot (b+1)}\,. \tag{viii}$$

(Note that the formula in (viii) is an instantiation of $x = y \wedge y = z \;\rightarrow\; x = z$.) Finally, by (vi), (vii), (viii) and (PC), we have

$$N \vdash \overline{a} \cdot \overline{b+1} = \overline{a \cdot (b+1)}\,.$$

This completes the proof of Lemma 2.8.4(6).

The proof of Lemma 2.8.4(7) is similar to the proof Lemma 2.8.4(6). Apply the axioms N7 and N8 in place of, respectively, N5 and N6.

Section 2.8.1, Exercise 6, page 71:

We will derive $\neg S0 = SS0$ from the axioms of N. Let us carry out a derivation in all its gory details. We are sure you will be convinced that 1 is different from 2 after you have seen this.

1.	$(\forall x)(\forall y)[Sx = Sy \;\rightarrow\; x = y]$	N2
2.	$(\forall x)(\forall y)[Sx = Sy \;\rightarrow\; x = y] \;\rightarrow\; (\forall y)[SS0 = Sy \;\rightarrow\; S0 = y]$	(Q1)
3.	$(\forall y)[SS0 = y \;\rightarrow\; S0 = y]$	1, 2, (PC)
4.	$(\forall y)[SS0 = Sy \;\rightarrow\; S0 = y] \;\rightarrow\; [SS0 = S0 \;\rightarrow\; S0 = 0]$	(Q1)
5.	$SS0 = S0 \;\rightarrow\; S0 = 0$	3, 4, (PC)
6.	$(\forall x)\neg Sx = 0$	N1
7.	$(\forall x)[\neg Sx = 0] \;\rightarrow\; [\neg S0 = 0]$	(Q1)
8.	$\neg S0 = 0$	6, 7, (PC)
9.	$\neg SS0 = S0$	5, 8, (PC)

So far, so good. Now we have derived that 2 does not equal 1. But we want to derive that 1 does not equal 2. So there is more work to do:

10.	$x = y \wedge x = x \;\rightarrow\; (x = x \;\rightarrow\; y = x)$	(E3)
11.	$x = x$	(E1)
12.	$\neg y = x \;\rightarrow\; \neg x = y$	10, 11, (PC)
13.	$(0 = 0 \vee \neg 0 = 0) \;\rightarrow\; [\neg y = x \;\rightarrow\; \neg x = y]$	12, (PC)
14.	$(0 = 0 \vee \neg 0 = 0) \;\rightarrow\; (\forall y)[\neg y = x \;\rightarrow\; \neg x = y]$	13, (QR)
15.	$(0 = 0 \vee \neg 0 = 0) \;\rightarrow\; (\forall x)(\forall y)[\neg y = x \;\rightarrow\; \neg x = y]$	14, (QR)
16.	$(\forall x)(\forall y)[\neg y = x \;\rightarrow\; \neg x = y]$	15, (PC)
17.	$(\forall x)(\forall y)[\neg y = x \;\rightarrow\; \neg x = y] \;\rightarrow\; (\forall y)[\neg y = S0 \;\rightarrow\; \neg S0 = y]$	(Q1)
18.	$(\forall y)[\neg y = S0 \;\rightarrow\; \neg S0 = y]$	16, 17, (PC)
19.	$(\forall y)[\neg y = S0 \;\rightarrow\; \neg S0 = y] \;\rightarrow\; [\neg SS0 = S0 \;\rightarrow\; \neg S0 = SS0]$	(Q1)
20.	$\neg SS0 = S0 \;\rightarrow\; \neg S0 = SS0$	18, 19, (PC)
21.	$\neg S0 = SS0$	9, 20, (PC)

Section 2.8.1, Exercise 8, page 71:

We are asked to prove that N does not prove that addition is commutative. We will restrict ourselves to showing that the first four axioms of N do not prove that addition is commutative. So, let us pretend that $\mathcal{L}_{NT}$ is $\{0, S, +\}$ and that the axioms of N are

(N_1) $(\forall x)\neg Sx = 0$

(N_2) $(\forall x)(\forall y)[Sx = Sy \rightarrow x = y]$

(N_3) $(\forall x)x + 0 = x$

(N_4) $(\forall x)(\forall y)x + Sy = S(x + y)$.

We will provide a model $\mathfrak{A}$ such that

$$\mathfrak{A} \models \{N_1, N_2, N_3, N_4\} \text{ and } \mathfrak{A} \not\models (\forall x)(\forall y)x + y = y + x.$$

Then, it follows by the Soundness Theorem that $N \nvdash (\forall x)(\forall y)x + y = y + x$.

The universe of $\mathfrak{A}$ will be the natural numbers extended by two nonstandard numbers. We will denote these nonstandard numbers α and β. Thus, $A = \mathbb{N} \cup \{\alpha, \beta\}$.

Let $0^{\mathfrak{A}} = 0$, let

$$S^{\mathfrak{A}}(x) = \begin{cases} x+1 & \text{if } x \in \mathbb{N} \\ x & \text{if } x \in \{\alpha, \beta\} \end{cases}$$

and let

$$x +^{\mathfrak{A}} y = \begin{cases} x+y & \text{if } x, y \in \mathbb{N} \\ x & \text{if } x \in \{\alpha, \beta\} \text{ and } y \in \mathbb{N} \\ y & \text{otherwise.} \end{cases}$$

(Recall that functions symbols shall be interpreted as total functions. Thus, we have to take care such that $S^{\mathfrak{A}}(x)$ and $x +^{\mathfrak{A}} y$ are defined for any $x, y \in A$.)

Now, we have

$$\alpha +^{\mathfrak{A}} \beta \quad = \quad \beta \quad \neq \quad \alpha \quad = \quad \beta +^{\mathfrak{A}} \alpha$$

and thus $\mathfrak{A} \not\models (\forall x)(\forall y) x + y = y + x$. It is easy to see that $\mathfrak{A} \models \{N_1, N_2, N_3\}$. We need to argue that we also have $\mathfrak{A} \models N_4$, that is, we need to argue that the equation

$$x \ +^{\mathfrak{A}} \ S^{\mathfrak{A}}(y) \quad = \quad S^{\mathfrak{A}}(\ x +^{\mathfrak{A}} \ y) \qquad (*)$$

holds for all $x, y \in \mathbb{N} \cup \{\alpha, \beta\}$.

It is obvious that (*) holds when both x and y are in $\mathbb{N}$. Let us check the case when $x \in \{\alpha, \beta\}$ and $y \in \mathbb{N}$. Then, for the left hand side of (*), we have $x +^{\mathfrak{A}} S^{\mathfrak{A}}(y) = x$, and for the right hand side of (*), we have $S^{\mathfrak{A}}(x +^{\mathfrak{A}} y) = S^{\mathfrak{A}}(x) = x$, and thus (*) holds. Finally, check the case when $y \in \{\alpha, \beta\}$ (and $x \in \mathbb{N} \cup \{\alpha, \beta\}$): we have $x +^{\mathfrak{A}} S^{\mathfrak{A}}(y) = x +^{\mathfrak{A}} y = y$ and $S^{\mathfrak{A}}(x +^{\mathfrak{A}} y) = S^{\mathfrak{A}}(y) = y$, and thus, (*) holds.

Section 3.2.1

Section 3.2.1, Exercise 1, page 86:

By definition, Σ is inconsistent when $\Sigma \vdash \bot$ where $\bot :\equiv [(\forall x)(x = x)] \wedge \neg[(\forall x)(x = x)]$. Furthermore, $\{\bot\}_P = \{\bot_P\} = \{A \wedge \neg A\}$. Let ϕ be any formula. Now, by Definition 2.4.1, ϕ_P is a propositional consequence of $\{\bot\}_P$ if every truth assignment that makes each formula in $\{\bot\}_P$ true also make ϕ_P true. But no assignment makes each formula in $\{\bot\}_P$ true. (The only formula in the set is $A \wedge \neg A$, and this formula is false no matter which truth value we assign to A.) Hence, ϕ_P is a propositional consequence of $\{\bot\}_P$. Thus, if we can derive $\bot$, then we can also derive ϕ since we have the inference rule (PC) in our deductive system. Or, to put it another way, if your theory is inconsistent, then you can prove anything.

Section 3.2.1, Exercise 2, page 86:

Let $\Sigma_0, \Sigma_1, \Sigma_2, \subseteq \ldots$ be consistent sets of sentences such that $\Sigma_0 \subseteq \Sigma_1 \subseteq \Sigma_2 \subseteq \ldots$.

Assume for the sake of contradiction that the set $\bigcup \Sigma_i$ is inconsistent, that is, $\bigcup \Sigma_i \vdash \bot$ Derivations in our deductive system are finite. Thus, there must be a finite set Γ such that $\Gamma \subseteq \bigcup \Sigma_i$ and $\Gamma \vdash \bot$. Since Γ is finite, it must be the case $\Gamma \subseteq \Sigma_k$ when k is sufficiently large. But then $\Sigma_k \vdash \bot$ (when k is sufficiently large). This contradicts that Σ_k is consistent.

Section 3.2.1, Exercise 3, page 86:

Let Π be any consistent set of sentences and let σ be an arbitrary sentence. We will prove that either the set $\Pi \cup \{\sigma\}$, or the set $\Pi \cup \{\neg\sigma\}$, is consistent.

Assume that both sets are inconsistent, that is

$$\Pi \cup \{\sigma\} \vdash \bot \quad \text{and} \quad \Pi \cup \{\neg\sigma\} \vdash \bot .$$

By the Deduction Theorem, we have

$$\Pi \vdash \sigma \to \bot \quad \text{and} \quad \Pi \vdash \neg\sigma \to \bot .$$

Observe that $\bot$ follows tautologically from $\sigma \to \bot$ and $\neg\sigma \to \bot$. Thus, by (PC) we have $\Pi \vdash \bot$. This contradicts that Π is consistent.

This proves that either the set $\Pi \cup \{\sigma\}$, or the set $\Pi \cup \{\neg\sigma\}$, is consistent. Hence, if $\Pi \cup \{\sigma\}$ is inconsistent, then $\Pi \cup \{\neg\sigma\}$ is consistent.

Section 3.3.1

Section 3.3.1, Exercise 3, page 93:

Let $\mathfrak{A}$ and $\mathfrak{B}$ be $\mathcal{L}$-structures such that $\mathfrak{A} \cong \mathfrak{B}$, and let $\imath : A \to B$ denote an isomorphism. We will prove that $\mathfrak{A} \equiv \mathfrak{B}$.

Let $s : \mathit{Vars} \to A$ be an assignment function. Let $\widehat{s}(x) = \imath(s(x))$. Then, $\widehat{s} : \mathit{Vars} \to B$ is an assignment function.

We define the notation $t^{\mathfrak{C}}[s]$ inductively over the structure of the term t. For any $\mathcal{L}$-structure $\mathfrak{C}$ and any assignment function $s : \mathit{Vars} \to C$, let

- $x^{\mathfrak{C}}[s] = s(x)$ when $x \in \mathit{Vars}$
- $c^{\mathfrak{C}}[s] = c$ when $c^{\mathfrak{C}}$ is a constant
- $(ft_1 \ldots t_n)^{\mathfrak{C}}[s] = f^{\mathfrak{C}} t_1^{\mathfrak{C}}[s] \ldots t_n^{\mathfrak{C}}[s]$ when f is a function symbol of arity n and $t_1, \ldots, t_n$ are terms.

We will need the following claim.

(Claim) For any $\mathcal{L}$-term t and any assignment $s : \mathit{Vars} \to A$, we have

$$\imath(t^{\mathfrak{A}}[s]) \;=\; t^{\mathfrak{B}}[\widehat{s}]\,.$$

We prove this claim by induction over the build-up of the term t.

Case: *t* ***is a variable*** *x.* We have

$$\imath(x^{\mathfrak{A}}[s]) \;=\; \imath(s(x)) \;=\; \widehat{s}(x) \;=\; x^{\mathfrak{B}}[\widehat{s}]\,.$$

The second equality holds by the definition of $\widehat{s}$.

Case: *t* ***is a constant*** *c.* We have

$$\imath(c^{\mathfrak{A}}[s]) \;=\; \imath(c^{\mathfrak{A}}) \;=\; c^{\mathfrak{B}} \;=\; c^{\mathfrak{B}}[\widehat{s}]\,.$$

The second equality holds since $\imath$ is an isomorphism.

Case: *t* ***is in the form*** *$ft_1 \dots t_n$.* We have

$$\begin{aligned}&\imath((ft_1 \dots t_n)^{\mathfrak{A}}[s]) \;=\; \imath(f^{\mathfrak{A}} t_1^{\mathfrak{A}}[s] \dots t_n^{\mathfrak{A}}[s]) \;=\\ &f^{\mathfrak{B}}\imath(t_1^{\mathfrak{A}}[s]) \dots \imath(t_n^{\mathfrak{A}}[s]) \;=\; f^{\mathfrak{B}} t_1^{\mathfrak{B}}[\widehat{s}] \dots t_n^{\mathfrak{B}}[\widehat{s}] \;=\; (ft_1 \dots t_n)^{\mathfrak{B}}[\widehat{s}]\,.\end{aligned}$$

The second equality holds since $\imath$ is an isomorphism, and the third equality holds by the induction hypothesis. This completes the proof of our claim.

We will prove

$$\mathfrak{A} \models \phi[s] \quad\Leftrightarrow\quad \mathfrak{B} \models \phi[\widehat{s}] \tag{*}$$

for any $\mathcal{L}$-formula ϕ. It follows straightforwardly from (*) that $\mathfrak{A} \equiv \mathfrak{B}$.

We prove (*) by induction on the complexity of ϕ.

Case: *ϕ* ***is of the form*** *$=t_1 t_2$.* We have

$$\begin{aligned}\mathfrak{A} \models {=}t_1 t_2[s] \quad &\Leftrightarrow\quad t_1^{\mathfrak{A}}[s] = t_2^{\mathfrak{A}}[s] && \\ &\Leftrightarrow\quad \imath(t_1^{\mathfrak{A}}[s]) = \imath(t_2^{\mathfrak{A}}[s]) && \imath \text{ is a bijection} \\ &\Leftrightarrow\quad t_1^{\mathfrak{B}}[\widehat{s}] = t_2^{\mathfrak{B}}[\widehat{s}] && \text{(Claim)} \\ &\Leftrightarrow\quad \mathfrak{B} \models {=}t_1 t_2[\widehat{s}]\,. &&\end{aligned}$$

Case: ϕ ***is of the form*** $Rt_1 \dots t_n$**.** We can without loss of generality assume that R is a unary relation symbol. We have

$$\begin{aligned}
\mathfrak{A} \models Rt[s] \quad &\Leftrightarrow \quad t^{\mathfrak{A}}[s] \in R^{\mathfrak{A}} \\
&\Leftrightarrow \quad \imath(t^{\mathfrak{A}}[s]) \in R^{\mathfrak{B}} && \imath \text{ is an isomorphism} \\
&\Leftrightarrow \quad t^{\mathfrak{B}}[\widehat{s}] \in R^{\mathfrak{B}} && \text{(Claim)} \\
&\Leftrightarrow \quad \mathfrak{B} \models Rt[\widehat{s}] \, .
\end{aligned}$$

Case: ϕ ***is of the form*** $(\forall x)(\alpha)$**.** We have

$$\begin{array}{ll}
\mathfrak{A} \models (\forall x)(\alpha)[s] & \\
\Updownarrow & \\
\text{for each } a \in A,\ \mathfrak{A} \models \alpha[s[x|a]] & \\
\Updownarrow & \text{(ind. hyp.)} \\
\text{for each } a \in A,\ \mathfrak{B} \models \alpha[\widehat{s[x|a]}] & \\
\Updownarrow & (\imath \text{ is a bijection}) \\
\text{for each } b \in B,\ \mathfrak{B} \models \alpha[\widehat{s}[x|b]] & \\
\Updownarrow & \\
\mathfrak{B} \models (\forall x)(\alpha)[\widehat{s}] \, . &
\end{array}$$

Case: ϕ ***is of the form*** $(\neg\alpha)$**.**

$$\begin{aligned}
\mathfrak{A} \models (\neg\alpha)[s] \quad &\Leftrightarrow \quad \mathfrak{A} \not\models \alpha[s] \\
&\Leftrightarrow \quad \mathfrak{B} \not\models \alpha[\widehat{s}] && \text{ind. hyp.} \\
&\Leftrightarrow \quad \mathfrak{B} \models (\neg\alpha)[\widehat{s}] \, .
\end{aligned}$$

Case: ϕ ***is of the form*** $(\alpha \vee \beta)$**.**

$$\begin{aligned}
\mathfrak{A} \models (\alpha \vee \beta)[s] \quad &\Leftrightarrow \quad \mathfrak{A} \models \alpha[s] \text{ or } \mathfrak{A} \models \beta[s] \\
&\Leftrightarrow \quad \mathfrak{B} \models \alpha[\widehat{s}] \text{ or } \mathfrak{B} \models \beta[\widehat{s}] && \text{ind. hyp.} \\
&\Leftrightarrow \quad \mathfrak{B} \models (\alpha \vee \beta)[\widehat{s}]
\end{aligned}$$

This completes the proof of (*).

Section 3.3.1, Exercise 5, page 93:
This exercise is solved as Exercise 6 below. Let $\phi(x)$ in Exercise 6 be the formula

$$(\forall y)(\forall z)[\, y \cdot z = x \ \rightarrow \ (y = S0 \ \vee z = S0)\,] \, .$$

Section 3.3.1, Exercise 6, page 93:

Let the $\mathfrak{N}^*$ be a nonstandard model of arithmetic, that is, an $\mathcal{L}_{NT}$-structure that is elementary equivalent, but not isomorphic, to the standard model $\mathfrak{N}$.

Assume there exists infinitely many $a \in \mathbb{N}$ such that $\mathfrak{N} \models \phi(x)[s[x|a]]$. Then, we have

$$\mathfrak{N} \models (\forall y)(\exists x)[y < x \ \wedge \ \phi(x)] \ .$$

Since $\mathfrak{N}^*$ is elementarily equivalent to $\mathfrak{N}$, we also have

$$\mathfrak{N}^* \models (\forall y)(\exists x)[y < x \ \wedge \ \phi(x)] \ .$$

Let c be an infinite number of $\mathfrak{N}^*$. Then

$$\mathfrak{N}^* \models (\exists x)[y < x \ \wedge \ \phi(x)] \ [s[y|c]]$$

and thus, there exists b such that

$$\mathfrak{N}^* \models y < x \ \wedge \ \phi(x) \ [s[y|c][x|b]] \ .$$

This b is an infinite number and $\mathfrak{N}^* \models \phi(x)[s[x|b]]$.

Section 3.3.1, Exercise 8, page 93:

(a) For $n \geq 2$, we define the formula σ'_n inductively by

- $\sigma'_2 \ :\equiv \ x_1 \neq x_2$
- $\sigma'_{n+1} \ :\equiv \ \sigma'_n \wedge x_1 \neq x_{n+1} \wedge x_2 \neq n_{n+1} \wedge \ldots \wedge x_n \neq n_{n+1}$.

Let $\sigma_n :\equiv (\exists x_1) \ldots (\exists x_n)[\sigma'_n]$. If $\mathfrak{A} \models \sigma_n$, then there will be at least n elements in the universe of $\mathfrak{A}$. Let $\Sigma = \{\sigma_2, \sigma_3, \ldots\}$. Then, every model of Σ is infinite.

(b) Assume that Γ is a set with arbitrarily large finite models. (We will prove that Γ has an infinite model.) Let $\mathfrak{A}_k$ denote a model of Γ that has at least k elements in its universe. Thus, there exists arbitrarily large $k \in \mathbb{N}$ such that $\mathfrak{A}_k \models \Gamma$.

Let Δ be any finite subset of $\Gamma \cup \Sigma$ where Σ is the set from (a). Since Δ is finite, there will be a largest n such that $\sigma_n \in \Delta$. Now, pick a model $\mathfrak{A}_k$ where $k \geq n$. Then we have $\mathfrak{A}_k \models \Delta$. This proves that any finite subset of $\Gamma \cup \Sigma$ has a model.

We are now ready to apply the Compactness Theorem: We know that any finite subset of $\Gamma \cup \Sigma$ has a model, and then $\Gamma \cup \Sigma$ has a model. A model of $\Gamma \cup \Sigma$ is of course also a model for Γ. Moreover, a model for $\Gamma \cup \Sigma$ is infinite because it is a model for Σ. (We know from (a) that any model of Σ is infinite.) Hence, we conclude that Γ has an infinite model.

(d) Assume that we for any structure $\mathfrak{A}$ we have

$$\mathfrak{A} \models \Phi \quad \Leftrightarrow \quad A \text{ is infinite} \qquad (*)$$

where A is the universe of $\mathfrak{A}$ and Φ is a finite set $\{\phi_1, \ldots, \phi_n\}$. Then we also have

$$\mathfrak{A} \models \neg(\phi_1 \wedge \ldots \wedge \phi_n) \quad \Leftrightarrow \quad A \text{ is finite .}$$

Thus, the set $\{\neg(\phi_1 \wedge \ldots \wedge \phi_n)\}$ has arbitrarily large finite models. By exercise (b), we know that $\{\neg(\phi_1 \wedge \ldots \wedge \phi_n)\}$ also has an infinite model $\mathfrak{B}$. It is easy to see that $\mathfrak{B} \not\models \Phi$. Hence, we have a structure with an infinite universe that is not a model of Φ. This contradicts (*).

Section 3.3.1, Exercise 9, page 94:

Suppose that Σ_1 and Σ_2 are two set of sentences such that no structure is a model of both Σ_1 and Σ_2. (We will show that there is a sentence α such that every model of Σ_1 is a model of α, and furthermore, every model of Σ_2 is a model of $\neg\alpha$.)

The set $\Sigma_1 \cup \Sigma_2$ has no model, and by the Compactness Theorem there exists a finite subset of $\Sigma_1 \cup \Sigma_2$ that has no model. Let

$$\Sigma = \{\alpha_1, \ldots, \alpha_n, \beta_1, \ldots \beta_m\}$$

be such a finite subset where $\{\alpha_1, \ldots, \alpha_n\} \subseteq \Sigma_1$ and $\{\beta_1, \ldots, \beta_m\} \subseteq \Sigma_2$. Furthermore, let $\alpha :\equiv \alpha_1 \wedge \ldots \wedge \alpha_n$. We claim that that

(i) every model of of Σ_1 is also a model of α

(ii) every model of of Σ_2 is also a model of $\neg\alpha$.

It is easy to see that (i) holds as $\{\alpha_1, \ldots, \alpha_n\}$ is a subset of Σ_1. Now, suppose that (ii) does not hold. Then, there exists a structure $\mathfrak{A}$ such that $\mathfrak{A} \models \Sigma_2$ and $\mathfrak{A} \models \alpha$. But then we also have $\mathfrak{A} \models \{\alpha_1, \ldots, \alpha_n, \beta_1, \ldots \beta_m\}$, and this contradicts our assumption that no structure is a model for $\{\alpha_1, \ldots, \alpha_n, \beta_1, \ldots \beta_m\}$. Thus, (ii) holds.

Section 3.3.1, Exercise 10, page 94:

We will show that there is no set of $\mathcal{L}$-sentences Σ such that

(i) Σ has an infinite model $\mathfrak{A}$ in which $<^{\mathfrak{A}}$ is a linear order of A

(ii) if $\mathfrak{B} \models \Sigma$, then $<^{\mathfrak{B}}$ is a well-ordering of B.

Now, suppose that (i) is satisfied. We will prove that there exists a model $\mathfrak{B}$ of Σ where $<^{\mathfrak{B}}$ is a not well-ordering of B. (Hence, (ii) is not satisfied.)

Let $c_0, c_1, c_2, \ldots$ be fresh constant symbols, that is, constant symbols that are not in the language $\mathcal{L}$, and let $\mathcal{L}_c$ be the language $\mathcal{L}$ extended with these constant symbols. Furthermore, let

$$\Gamma \;=\; \Sigma \,\cup\, \{c_{i+1} < c_i \mid i \in \mathbb{N}\}\,.$$

We will argue that every finite subset of Γ has a model.

Pick an arbitrary finite subset Γ_0 of Γ. There will be a largest n such that the formula $c_n < c_{n-1}$ is in Γ_0. Fix this n. (Let n be 0 if there are no sentences of this form in Γ_0.) Now, $<^{\mathfrak{A}}$ is a linear order of the infinite set A. Thus, there exists a $<^{\mathfrak{A}}$-descending chain of length n, that is, we have $a_1, \ldots, a_n \in A$ such that

$$a_1 \;>^{\mathfrak{A}}\; a_2 \;>^{\mathfrak{A}}\; \ldots \;>^{\mathfrak{A}}\; a_n\,.$$

We define the $\mathcal{L}_c$-structure $\mathfrak{C}$ by extending the $\mathcal{L}$-structure $\mathfrak{A}$ with

- $c_i^{\mathfrak{C}} = a_i$ for $i = 1, \ldots n$
- $c_i^{\mathfrak{C}} = a$ if $i > n$ (where a is an arbitrary element in A).

It is easy to see that $\mathfrak{C}$ is a model for Γ_0. This proves that any finite set of Γ has a model. By the Compactness Theorem, we conclude that Γ has a model $\mathfrak{C}$. For any $i \in \mathbb{N}$, we find the sentence $c_{i+1} < c_i$ in Γ. Hence

$$c_0^{\mathfrak{C}} \;>^{\mathfrak{C}}\; c_1^{\mathfrak{C}} \;>^{\mathfrak{C}}\; c_2^{\mathfrak{C}} \;>^{\mathfrak{C}}\; \ldots$$

and we conclude that $<^{\mathfrak{C}}$ is not a well-ordering of the universe. Moreover, the $\mathcal{L}_c$-structure $\mathfrak{C}$ is a model of Σ as $\Sigma \subseteq \Gamma$. Let $\mathfrak{B} = \mathfrak{C}_{|\mathcal{L}}$. Then the $\mathcal{L}$-structure $\mathfrak{B}$ is an infinite model of Σ, and $<^{\mathfrak{B}}$ is not a well-ordering of B.

Section 3.3.1, Exercise 11, page 94:

Let $\mathfrak{N}^*$ be any nonstandard model of arithmetic. We will show that $<$ is not a well-order in $\mathfrak{N}^*$

Let $\exists! y[\phi(y)]$ abbreviate $\exists y[\phi(y) \wedge \forall z[\phi(z) \rightarrow z = y]]$. Now, $\mathfrak{A} \models \exists! y[\phi(y)][s]$ if, and only if, there exists a unique $a \in A$ such that $\mathfrak{A} \models \phi(y)[s[y|a]]$.

We have

$$\mathfrak{N} \models \forall x[\, x \neq 0 \;\rightarrow\; \exists! y[x = Sy]\,]\,.$$

Since $\mathfrak{N}$ and $\mathfrak{N}^*$ are elementary equivalent, we also have

$$\mathfrak{N}^* \models \forall x[\, x \neq 0 \;\rightarrow\; \exists! y[x = Sy]\,]\,. \qquad (*)$$

Let a be a nonstandard element of $\mathfrak{N}^*$, and let s be an assignment function. It follows from (*) that

$$\mathfrak{N}^* \models x \neq 0 \;\rightarrow\; \exists! y[x = Sy]\;[s[x|a]]\,. \qquad (**)$$

Since $\mathfrak{N}^* \models x \neq 0[s[x|a]]$ for any nonstandard a, it follows from (**) that

$$\mathfrak{N}^* \models \exists! y[x = Sy]\ [s[x|a]]\ .$$

Thus, for any nonstandard a, there exists a unique b such that $\mathfrak{N}^* \models x = Sy[s[x|a][y|b]]$. Let $p(a)$ denote the one and only b such that $\mathfrak{N}^* \models x = Sy[s[x|a][y|b]]$.

We have $S^{\mathfrak{N}^*}(p(a)) = a$ for any nonstandard number a. Furthermore, $\mathfrak{N} \models (\forall x)x < Sx$. By elementary equivalence, $\mathfrak{N}^* \models (\forall x)x < Sx$. This entails that $p(a) <^{\mathfrak{N}^*} a$ holds for any nonstandard number a.

Now, the number $p(a)$ will of course be nonstandard when a is. Thus, let a be any nonstandard number, and

$$a \quad >^{\mathfrak{N}^*} \quad p(a) \quad >^{\mathfrak{N}^*} \quad p(p(a)) \quad >^{\mathfrak{N}^*} \quad p(p(p(a))) \quad >^{\mathfrak{N}^*} \quad \ldots$$

is an infinite descending chain. A well-order does not have an infinite descending chain, and we can conclude that $<$ is not a well-order in $\mathfrak{N}^*$.

Section 3.4.1

Section 3.4.1, Exercise 2, page 101:
Assume that we have

$$\mathfrak{A} \models \phi[s] \quad \Rightarrow \quad \mathfrak{B} \models \phi[s] \tag{*}$$

for any formula ϕ and any assignment function $s : \mathit{Vars} \to A$. We will prove that we also have

$$\mathfrak{B} \models \phi[s] \quad \Rightarrow \quad \mathfrak{A} \models \phi[s] \tag{**}$$

for any formula ϕ any assignment function $s : \mathit{Vars} \to A$.

Pick an arbitrary formula ϕ and an arbitrary assignment function $s : \mathit{Vars} \to A$. By (*), we have

$$\mathfrak{A} \models \neg\phi[s] \quad \Rightarrow \quad \mathfrak{B} \models \neg\phi[s]\ . \tag{i}$$

By (i) and Definition 1.7.4, we have

$$\mathfrak{A} \not\models \phi[s] \quad \Rightarrow \quad \mathfrak{B} \not\models \phi[s]\ . \tag{ii}$$

Now, (ii) and (**) are equivalent as (ii) is the contrapositive version of (**).

Section 3.4.1, Exercise 3, page 101:
We assume (i) $\mathfrak{A} \prec \mathfrak{B}$, (ii) $\mathfrak{C} \prec \mathfrak{B}$ and (iii) $\mathfrak{A} \subseteq \mathfrak{C}$. We will prove $\mathfrak{A} \prec \mathfrak{C}$.

Fix an arbitrary formula ϕ. By (i), we have

$$\mathfrak{A} \models \phi[s] \quad \Leftrightarrow \quad \mathfrak{B} \models \phi[s] \tag{iv}$$

for every assignment function $s : \mathit{Vars} \to A$. By (ii), we have

$$\mathfrak{C} \models \phi[s] \quad \Leftrightarrow \quad \mathfrak{B} \models \phi[s] \tag{v}$$

for every assignment function $s : \mathit{Vars} \to C$.

By (iii), we have $A \subseteq C$. Thus, any assignment function from *Vars* into A is also an assignment function from *Vars* into C, and thus, by (iv) and (v), we have

$$\mathfrak{A} \models \phi[s] \quad \Leftrightarrow \quad \mathfrak{C} \models \phi[s] \tag{vi}$$

for every assignment function $s : \mathit{Vars} \to A$. Finally, by (iii) and (vi), we have $\mathfrak{A} \prec \mathfrak{C}$.

Section 3.4.1, Exercise 4, page 101:

Let

$$\mathfrak{A}_1 \prec \mathfrak{A}_2 \prec \mathfrak{A}_3 \prec \mathfrak{A}_4 \prec \dots \tag{i}$$

and let

$$\mathfrak{A} = \bigcup_{i=1}^{\infty} \mathfrak{A}_i \,. \tag{ii}$$

We will prove $\mathfrak{A}_i \prec \mathfrak{A}$ (for each i).

By Definition 3.4.4, we have to prove $\mathfrak{A}_i \subseteq \mathfrak{A}$ and

$$\mathfrak{A}_i \models \phi[s] \quad \Leftrightarrow \quad \mathfrak{A} \models \phi[s] \tag{iii}$$

for any $i \geq 1$, any formula ϕ, and any assignment function $s : \mathit{Vars} \to A_i$. It is easy to check that $\mathfrak{A}_i \subseteq \mathfrak{A}$ holds, and we omit the details. We will prove (iii) by induction on the complexity of ϕ.

First, we prove (iii) when ϕ is in the form $Rt_1, \dots, t_n$ (where R might be the equality relation). We have

$$\begin{array}{rl}
\mathfrak{A}_i \models Rt_1 \dots t_n[s] & \\
\Updownarrow & \text{(Def. 1.7.4)} \\
\langle \bar{s}(t_1), \dots, \bar{s}(t_n) \rangle \in R^{\mathfrak{A}_i} & \\
\Updownarrow & \text{(since } \mathfrak{A}_i \subseteq \mathfrak{A} \text{ and } s : \mathit{Vars} \to A_i\text{)} \\
\langle \bar{s}(t_1), \dots, \bar{s}(t_n) \rangle \in R^{\mathfrak{A}} & \\
\Updownarrow & \text{(Def. 1.7.4)} \\
\mathfrak{A} \models Rt_1 \dots t_n[s] \,. &
\end{array}$$

It is straightforward to prove (iii) in the case when ϕ is in the form $\neg\psi$:

$$\mathfrak{A}_i \models \neg\psi \quad \Leftrightarrow \quad \mathfrak{A}_i \not\models \psi \quad \Leftrightarrow \quad \mathfrak{A} \not\models \psi \quad \Leftrightarrow \quad \mathfrak{A} \models \neg\psi \,.$$

The second equivalence holds by the induction hypothesis. The first and the third equivalence hold by Definition 1.7.4. The case when ϕ is in the form $\alpha \vee \beta$ is also straightforward, and we omit the details.

Finally we prove (iii) when ϕ in the form $(\exists x)\psi$. (This case replaces the case for $(\forall x)\psi$. A proof for the case $(\forall x)\psi$ is similar but a bit harder to follow.) Observe that (ii) implies that

$$a \in A \quad \Leftrightarrow \quad a \in A_k \text{ for all sufficiently large } k \,. \qquad \text{(iii)}$$

Thus, when k is sufficiently large and $s : \mathit{Vars} \to A_i$ (let $k \geq i$), we have

$$\begin{array}{rl}
\mathfrak{A} \models (\exists x)\psi[s] & \\
\Updownarrow & \text{(Def. 1.7.4)} \\
\text{exists } a \in A \text{ such that } \mathfrak{A} \models \psi[s[x|a]] & \\
\Updownarrow & \text{(iii)} \\
\text{exists } a \in A_k \text{ such that } \mathfrak{A} \models \psi[s[x|a]] & \\
\Updownarrow & \text{(ind. hyp.)} \\
\text{exists } a \in A_k \text{ such that } \mathfrak{A}_k \models \psi[s[x|a]] & \\
\Updownarrow & \text{(Def. 1.7.4)} \\
\mathfrak{A}_k \models (\exists x)\psi[s] & \\
\Updownarrow & \text{(i)} \\
\mathfrak{A}_i \models (\exists x)\psi[s] &
\end{array}$$

This proves (iii) in the case when ϕ is in the form $(\exists x)\psi$.

Section 3.4.1, Exercise 6, page 101:

Let $\mathfrak{A} \prec \mathfrak{B}$. Furthermore, fix $b \in B$ such that we have

(i) $\mathfrak{B} \models \phi[s[x|b]]$

(ii) $\mathfrak{B} \not\models \phi[s[x|\hat{b}]]$ for every $\hat{b}$ in B that is different from b

for every $s : \mathit{Vars} \to B$. We will prove that $b \in A$.

Fix an assignment function s from *Vars* into A. Now, s is also an assignment function from *Vars* into B. Thus, by (i), we have $\mathfrak{B} \models (\exists x)\phi[s]$. Furthermore, since $\mathfrak{A} \prec \mathfrak{B}$, we have $\mathfrak{A} \models (\exists x)\phi[s]$. So, there exists $a \in A$ such that $\mathfrak{A} \models \phi[s[x|a]]$. Since $\mathfrak{A} \prec \mathfrak{B}$, we have $\mathfrak{B} \models \phi[s[x|a]]$. By (ii), b has to be the same element as a. This proves that $b \in A$.

Section 3.4.1, Exercise 9, page 101:

Let $\mathcal{L}$ be a first-order language, and let $\mathfrak{A}$ be an $\mathcal{L}$-structure. Let $\mathcal{L}(A)$ be the first-order language we get when $\mathcal{L}$ is extended by a new constant symbol $\overline{a}$ for each $a \in A$. Let $\overline{\mathfrak{A}}$ be the $\mathcal{L}(A)$-structure we get when $\mathfrak{A}$ is extended to interpret each new constant symbol $\overline{a}$ as a, that is, $\overline{a}^{\overline{\mathfrak{A}}} = a$. (So $\mathfrak{A}$ and $\overline{\mathfrak{A}}$ has the same universe.) Let

$$Th(\overline{\mathfrak{A}}) \;=\; \{\sigma \mid \sigma \text{ is an } \mathcal{L}(A)\text{-formula such that } \overline{\mathfrak{A}} \models \sigma\} \,.$$

(The set $Th(\overline{\mathfrak{A}})$ is called the *complete diagram of* $\mathfrak{A}$.) Finally, let $\overline{\mathfrak{B}}$ be any $\mathcal{L}(A)$-structure such that $\overline{\mathfrak{B}} \models Th(\overline{\mathfrak{A}})$, and let $\mathfrak{B} = \overline{\mathfrak{B}}|\mathcal{L}$.

We define the function $h : A \to B$ by $h(a) = \overline{a}^{\overline{\mathfrak{B}}}$ (for any $a \in A$). Let C be the range of h.

> **(Claim I)** The set C is closed under the function $f^{\mathfrak{B}}$ (for any function symbol f of $\mathcal{L}$).

Let $c \in C$. (We will prove that $f^{\mathfrak{B}}(c) \in C$.) As C is the range of h, we have $h(a) = \overline{a}^{\overline{\mathfrak{B}}} = c$ for some $a \in A$. There will be an element b in A such that the formula $f(\overline{a}) = \overline{b}$ is in $Th(\overline{\mathfrak{A}})$. As $\overline{\mathfrak{B}} \models Th(\overline{\mathfrak{A}})$, we have $f^{\mathfrak{B}}(\overline{a}^{\overline{\mathfrak{B}}}) = \overline{b}^{\overline{\mathfrak{B}}}$. Thus, we have $b \in A$, such that

$$f^{\mathfrak{B}}(c) \;=\; f^{\mathfrak{B}}(\overline{a}^{\overline{\mathfrak{B}}}) \;=\; \overline{b}^{\overline{\mathfrak{B}}} \;=\; h(b) \,.$$

This proves that $f^{\mathfrak{B}}(c)$ is in the range of h, and the range of h equals C. The proof can easily be generalized to functions of arbitrary arity. Hence, we conclude that (Claim I) holds.

We define $\mathfrak{C}$ as the substructure of $\mathfrak{B}$ that has universe C. The structure $\mathfrak{C}$ is well defined by (Claim I).

> **(Claim II)** The function h is an isomorphism between $\mathfrak{A}$ and $\mathfrak{C}$.

To prove (Claim II), we must prove that $h : A \to C$ is a bijection such that

(i) $h(c^{\mathfrak{A}}) = c^{\mathfrak{C}}$ for any constant symbol c of $\mathcal{L}$

(ii) $h(f^{\mathfrak{A}}(a_1, \ldots, a_n)) = f^{\mathfrak{C}}(h(a_1), \ldots, h(a_n))$ for any function symbol f of $\mathcal{L}$ and any $a_1, \ldots a_n \in A$

(iii) $\langle a_1, \ldots, a_n\rangle \in R^{\mathfrak{A}} \;\Leftrightarrow\; \langle h(a_1), \ldots, h(a_n)\rangle \in R^{\mathfrak{C}}$ for any relation symbol R of $\mathcal{L}$ and any $a_1, \ldots a_n \in A$.

The following argument shows that h also is an injection: Pick $a, b \in A$ such that $a \neq b$. The formula $\neg\overline{a} = \overline{b}$ will be in the set $Th(\overline{\mathfrak{A}})$. Since $\overline{\mathfrak{B}} \models Th(\overline{\mathfrak{A}})$, we have $h(a) = \overline{a}^{\overline{\mathfrak{B}}} \neq \overline{b}^{\overline{\mathfrak{B}}} = h(b)$. This proves that h is

an injection. It is trivial that h is a surjection as C is the range of h. Thus, h is a bijection.

We will now prove (i). Let c be a constant symbol of $\mathcal{L}$. For some $a \in A$, the formula $c = \overline{a}$ will be in the set $Th(\overline{\mathfrak{A}})$. Thus, as $\overline{\mathfrak{A}} \models Th(\overline{\mathfrak{A}})$, we have $c^{\mathfrak{A}} = \overline{a}^{\mathfrak{A}} = a$, and as $\overline{\mathfrak{B}} \models Th(\overline{\mathfrak{A}})$, we have $c^{\overline{\mathfrak{B}}} = \overline{a}^{\overline{\mathfrak{B}}}$. Hence,

$$h(c^{\mathfrak{A}}) \;=\; h(a) \;=\; \overline{a}^{\overline{\mathfrak{B}}} \;=\; c^{\overline{\mathfrak{B}}} \;=\; c^{\mathfrak{C}}\,.$$

The second equality holds by the definition of h, and the last equality holds as $\mathfrak{C}$ is a substructure of $\mathfrak{B}$. This proves that (i) holds.

We turn to the proof of (ii) Let f be a function symbol of $\mathcal{L}$. We can without loss of generality assume that f is a unary function symbol. Pick an arbitrary $a \in A$, and let b be the unique element of A such that $f^{\mathfrak{A}}(a) = b$. Now, the formula $f(\overline{a}) = \overline{b}$ will be in the set $Th(\overline{\mathfrak{A}})$. Since $\overline{B} \models Th(\overline{\mathfrak{A}})$, we have $f^{\overline{\mathfrak{B}}}(\overline{a})^{\overline{\mathfrak{B}}} = \overline{b}^{\overline{\mathfrak{B}}}$. Thus,

$$\begin{aligned}
h(f^{\mathfrak{A}}(a)) &= h(b) && \text{(by choice of } b\text{)}\\
&= \overline{b}^{\overline{\mathfrak{B}}} && \text{(def. of } h\text{)}\\
&= f^{\overline{\mathfrak{B}}}(\overline{a}^{\overline{\mathfrak{B}}}) && \text{(since } f^{\overline{\mathfrak{B}}}(\overline{a})^{\overline{\mathfrak{B}}} = \overline{b}^{\overline{\mathfrak{B}}}\text{)}\\
&= f^{\overline{\mathfrak{B}}}(h(a)) && \text{(def. of } h\text{)}\\
&= f^{\mathfrak{B}}(h(a)) && \text{(as } \mathfrak{B} = \overline{B}|\mathcal{L}\ \text{)}\\
&= f^{\mathfrak{C}}(h(a))\,. && \text{(as } \mathfrak{C} \subseteq \mathfrak{B}\ \text{)}
\end{aligned}$$

This proves (ii).

The proof of (iii) is similar to the proof of (ii), and we skip the details. This completes the proof of (Claim II).

Next, we will prove

(Claim III) $\mathfrak{C} \prec \mathfrak{B}$.

We do already know that $\mathfrak{C} \subseteq \mathfrak{B}$. Thus, by Lemma 3.4.7, it will be sufficient to prove that

> **(*)** for every $\mathcal{L}$-formula α and every $s : \mathit{Vars} \to C$ such that $\mathfrak{B} \models (\exists x)\alpha[s]$, there is an $a \in C$ such that $\mathfrak{B} \models \alpha[s[x|a]]$.

In order to prove (*), we assume that $\mathfrak{B} \models (\exists x)\alpha(x, y)[s]$ where $s : \mathit{Vars} \to C$. We display the free variables in α, and we can without loss of generality assume that there is no other free variables than x and y. Let $b = h^{-1}(s(y))$. Then, we have $b \in A$. Moreover, the $\mathcal{L}(\mathfrak{A})$-formula $(\exists x)\alpha(x, \overline{b})$ will be in the set $Th(\overline{\mathfrak{A}})$. (Why is the formula in the set? Well, assume that this is not the case. Then $\neg(\exists x)\alpha(x, \overline{b})$

will be in $Th(\overline{\mathfrak{A}})$. But, then $\overline{\mathfrak{B}} \models \neg(\exists x)\alpha(x, \overline{b})$. This contradicts that $\mathfrak{B} \models (\exists x)\alpha(x, y)[s].$) Now, since $(\exists x)\alpha(x, \overline{b})$ is in $Th(\overline{\mathfrak{A}})$, the formula $\alpha(\overline{a}, \overline{b})$ will also be in $Th(\overline{\mathfrak{A}})$ for some $a \in A$. Thus $\overline{\mathfrak{B}} \models \alpha(\overline{a}, \overline{b})$. Thus

$$\overline{\mathfrak{B}} \;\models\; \alpha(x, y)\; [\, s\, [x|\overline{a}^{\overline{\mathfrak{B}}}]\, [y|\overline{b}^{\overline{\mathfrak{B}}}]\,]\; .$$

Observe that $\overline{a}^{\overline{\mathfrak{B}}} = h(a) \in C$ and $\overline{b}^{\overline{\mathfrak{B}}} = h(b) = s(y)$. Thus

$$\overline{\mathfrak{B}} \;\models\; \alpha(x, y)\; [\, s\, [x|h(a)][y|s(y)]\,]\; .$$

Thus

$$\overline{\mathfrak{B}} \;\models\; \alpha(x, y)\; [\, s\, [x|h(a)]\,]\; .$$

Thus

$$\mathfrak{B} \;\models\; \alpha(x, y)\; [\, s\, [x|h(a)]\,]$$

where $h(a) \in C$.

Section 3.4.1, Exercise 10, page 102:

We will prove Theorem 3.4.13 (Upward Löwenheim-Skolem Theorem.)

Let $\mathcal{L}$ be a countable language, let $\mathfrak{A}$ be an infinite $\mathcal{L}$-structure, and let K be a set of some infinite cardinality κ. We will show that $\mathfrak{A}$ has an elementary extension $\mathfrak{B}$ such that the cardinality of $\mathfrak{B}$ is greater than or equal to κ. (When we say that a model is of a certain cardinality, we mean that the universe of the model is of that cardinality.)

For each $k \in K$, we introduce a constant symbol c_k that is not in the language $\mathcal{L}(A)$. We will say that these constant symbols are *new*. Let $\mathcal{L}(A)_c = \mathcal{L}(A) \cup \{c_k \mid k \in K\}$. (You find the definition of $\mathcal{L}(A)$ in Exercise 9.) Let

$$\Sigma \;=\; Th(\overline{\mathfrak{A}}) \;\cup\; \{\neg c_i = c_j \mid i, j \in K \text{ and } i \neq j\}$$

where $Th(\overline{\mathfrak{A}})$ is the complete diagram of $\mathfrak{A}$. (You find the definition of $Th(\overline{\mathfrak{A}})$ in Exercise 9.) Now, Σ is a set of $\mathcal{L}(A)_c$-formulas, and we will now argue that any finite subset of Σ has a model.

Take a finite subset Σ' of Σ. Now, for some $n \in \mathbb{N}$,

- just n of the new constant symbols c_k occur in formulas of Σ'
- each of these n new constant symbols will only occur in formulas of the form $\neg c_i = c_j$ (and not occur in any formula taken from $Th(\overline{\mathfrak{A}})$)
- there is more than n elements in the universe of $\mathfrak{A}$ since $\mathfrak{A}$ is an infinite structure.

Hence, we can extend $\mathfrak{A}$ to an $\mathcal{L}(A)_c$-structure that is a model of Σ' by taking n distinct elements of $\mathfrak{A}$ and interpreting each of the n new constant symbols as one of these elements.

This argument shows that any finite subset of Σ has a model. By the Compactness Theorem, we conclude that Σ has a model $\overline{\mathfrak{B}}$. The cardinality of of $\overline{\mathfrak{B}}$ will be be at least κ since each of the new constant symbols corresponds to a unique element in the universe. Moreover, $\overline{\mathfrak{B}} \models Th(\overline{\mathfrak{A}})$. (This is obvious as $Th(\overline{\mathfrak{A}})$ is a subset of Σ.)

Let $\mathfrak{B}_0 = \overline{\mathfrak{B}}|\mathcal{L}$. Now, the cardinality of $\mathfrak{B}_0$ is the same as the cardinality of $\overline{\mathfrak{B}}$, and by Exercise 9, we can find a structure $\mathfrak{C}$ such that $\mathfrak{A} \cong \mathfrak{C} \prec \mathfrak{B}_0$. Now it is easy to construct a structure $\mathfrak{B}$ of the desired cardinality such that $\mathfrak{A} \prec \mathfrak{B}$. (Just rename some of the elements in $\mathfrak{B}$. We leave the details to the reader.)

Section 3.4.1, Exercise 11, page 102:

Let $\mathfrak{A}_c$ be a structure for the language $\mathcal{L}_{\mathbb{R}} \cup \{c\}$ such that

$$\mathfrak{A}_c \;\models\; Th(\mathfrak{R}) \;\cup\; \{\, \dot{0} \dot{<} c \,\} \;\cup\; \{\, c \dot{<} \dot{r} \mid r \in \mathbb{R} \text{ and } r > 0 \,\}$$

and let $\mathfrak{A} = \mathfrak{A}_c|\mathcal{L}_{\mathbb{R}}$. Now, $Th(\mathfrak{R})$ is nothing but the complete diagram of $\mathfrak{R}$, and $\mathfrak{A}$ is a model for $Th(\mathfrak{R})$. (Besides $\mathfrak{A}$ and $\mathfrak{A}|\mathcal{L}_{\mathbb{R}}$ are the same structure.) By Exercise 9, $\mathfrak{R}$ is isomorphic to an elementary substructure of $\mathfrak{A}$ or, to put it otherwise, there is an isomorphic copy of the real numbers living inside the structure $\mathfrak{A}$.

Section 5.2.1

Section 5.2.1, Exercise 1, page 119:

How can we prove that a sentence ϕ is not derivable from the axioms of N? Well, by the Soundness Theorem, we can do so by providing an $\mathcal{L}_{NT}$-structure $\mathfrak{A}$ such that $\mathfrak{A} \models N$ and $\mathfrak{A} \not\models \phi$.

We do already know that there exists $\mathfrak{A}$ such that

$$\mathfrak{A} \models N \quad \text{and} \quad \mathfrak{A} \not\models (\forall x)(\forall y) x + y \;=\; y + x \;.$$

(See the solution of Exercise 8 in Section 2.8.1) Thus, we have

$$N \nvdash (\forall x)(\forall y) x + y \;=\; y + x$$

by the Soundness Theorem. Now, does there exist an $\mathcal{L}_{NT}$-structure $\mathfrak{A}$ such that $\mathfrak{A} \models N$ and

$$\mathfrak{A} \not\models \neg[(\forall x)(\forall y) x + y \;=\; y + x] \;?$$

Yes, of course, the standard model $\mathfrak{N}$ is such a structure. We have

$$\mathfrak{N} \models N \quad \text{and} \quad \mathfrak{N} \not\models \neg[(\forall x)(\forall y) x + y \;=\; y + x]$$

and thus, by the Soundness Theorem, we also have

$$N \nvdash \neg[(\forall x)(\forall y)x + y \;=\; y + x] \; .$$

Section 5.2.1, Exercise 2, page 119:

Definition 3.3.4 states that

$$Th(\mathfrak{N}) \;=\; \{\phi \mid \phi \text{ is an } \mathcal{L}_{NT}\text{-formula and } \mathfrak{N} \models \phi\} \; .$$

Now, $\mathfrak{N} \models N$. Thus, by the Soundness Theorem, if $N \vdash \phi$, we also have $\mathfrak{N} \models \phi$. This entails that every formula which is derivable from the axioms of N, will be in the set $Th(\mathfrak{N})$, that is

$$N \vdash \phi \quad \Rightarrow \quad \phi \in Th(\mathfrak{N}) \; . \qquad (*)$$

Suppose that Σ is an axiomatization of $Th(\mathfrak{N})$, and suppose that $N \vdash \sigma$. According to Definition 4.2.1, we have $\Sigma \vdash \psi$ for any $\psi \in Th(\mathfrak{N})$. By (*), we have $\sigma \in Th(\mathfrak{N})$. Thus, $\Sigma \vdash \sigma$.

Section 5.2.1, Exercise 3, page 119:

Let $\mathfrak{A}$ be a nonstandard model of arithmetic. (So $\mathfrak{A}$ and the standard model $\mathfrak{N}$ are elementarily equivalent, but not isomorphic.) Definition 3.3.4 states that

$$Th(\mathfrak{A}) \;=\; \{\phi \mid \sigma \text{ is an } \mathcal{L}_{NT}\text{-formula and } \mathfrak{A} \models \phi\} \; .$$

Let σ be a sentence of $\mathcal{L}_{NT}$. We either have $\mathfrak{A} \models \sigma$, or we have $\mathfrak{A} \models \neg\sigma$, and then, either σ or $\neg\sigma$ will be a member of the set $Th(\mathfrak{A})$. If σ is in $Th(\mathfrak{A})$, then $Th(\mathfrak{A}) \vdash \sigma$. If $\neg\sigma$ is an element of $Th(\mathfrak{A})$, then $Th(\mathfrak{A}) \vdash \neg\sigma$. So we either have $Th(\mathfrak{A}) \vdash \sigma$ or we have $Th(\mathfrak{A}) \vdash \neg\sigma$. Thus, by Definition 4.1.1, $Th(\mathfrak{A})$ is complete theory.

Does $Th(\mathfrak{A})$ provides an axiomatization of $Th(\mathfrak{N})$? Well, according to Definition 4.1.2, the theory $Th(\mathfrak{A})$ is an axiomatization of $Th(\mathfrak{N})$ if every sentence in $Th(\mathfrak{N})$ is provable in the theory $Th(\mathfrak{A})$. And will every sentence in $Th(\mathfrak{N})$ be provable in this theory? Yes, indeed. This is trivially true because $\mathfrak{A}$ is elementarily equivalent to the standard model, and then, we have $Th(\mathfrak{A}) = Th(\mathfrak{N})$.

Section 5.3.1

Section 5.3.1, Exercise 1, page 128:

The Δ-formula $x = SSSSSSSSSSSSSSSS0$ defines the set $\{17\}$. So does the non-Δ-formula

$$(\forall y)[x \times x = y \;\rightarrow\; y = \overline{289}] \; .$$

Section 5.3.1, Exercise 2, page 128:

Let $A \subseteq \mathbb{N}$ be represented by $\phi(x)$. Let $B \subseteq \mathbb{N}$ be represented by $\psi(x)$.

(a) Let $\xi(x) :\equiv \phi(x) \vee \psi(x)$. We prove that $\psi(x)$ represents the set $A \cup B$.

Assume $a \in A \cup B$. (We will prove $N \vdash \xi(\overline{a})$.) We will find a in at least one of the sets A and B. If a is in A, then $N \vdash \phi(\overline{a})$ since $\phi(x)$ represents A. Now, $\phi(x) \vee \psi(x)$ follows tautologically from $\phi(x)$, and thus, we have $N \vdash \xi(\overline{a})$ by (PC). If a is not in A, then a is in B. Then, $N \vdash \psi(\overline{a})$ since $\psi(x)$ represents B. Now, $\phi(x) \vee \psi(x)$ follows tautologically from $\psi(x)$, and thus, by (PC), $N \vdash \xi(\overline{a})$.

Assume $a \notin A \cup B$. (We will prove $N \vdash \neg\xi(\overline{a})$.) Now we have $a \notin A$ and $a \notin B$. Then, $N \vdash \neg\phi(\overline{a})$ and $N \vdash \neg\psi(\overline{a})$ since $\phi(x)$ and $\psi(x)$ represent, respectively, A and B. Now, $\neg(\phi(\overline{a}) \vee \psi(\overline{a}))$ follows tautologically from $\neg\phi(\overline{a})$ and $\neg\psi(\overline{a})$. Hence, we have $N \vdash \neg\xi(\overline{a})$ by (PC).

We have proved that for any $a \in \mathbb{N}$

- $N \vdash \xi(\overline{a})$ if $a \in A \cup B$
- $N \vdash \neg\xi(\overline{a})$ if $a \notin A \cup B$.

By Definition 4.3.1 we conclude that the formula $\xi(x)$ represents the set $A \cup B$.

(b)

The formula $\phi(x) \wedge \psi(x)$ represents the set $A \cap B$. The proof of this is analogous to the proof in (a).

(c)

It is easy to prove that the formula $\neg\phi(x)$ represents the complement of the set A.

Section 5.3.1, Exercise 3, page 128:

Lemma 2.8.4(1) states that $N \vdash \overline{a} = \overline{b}$ if $a = b$. Lemma 2.8.4(2) states that $N \vdash \neg\overline{a} = \overline{b}$ if $a \neq b$. Thus, for any $n \in \mathbb{N}$, the formula $x = \overline{n}$ represents the singleton set $\{n\}$. Any finite set is a union of singleton sets, that is

$$\{a_1, \ldots, a_k\} \;=\; \{a_1\} \cup \ldots \cup \{a_k\}\,.$$

By Exercise 2(a), every finite set is representable. By Exercise 2(c), the complement of a finite set will also be representable.

Section 5.3.1, Exercise 7, page 128:

Let $A = \{\langle a, b\rangle \mid a \text{ divides } b\}$. Let $\mathit{Divides}(x, y)$ be the Δ-formula $(\exists z < y)[x \cdot z = y]$. Then, $\mathfrak{N} \models \mathit{Divides}(\overline{a}, \overline{b})$ if and only if a is a factor of b. Moreover, $\mathit{Divides}(x, y)$ defines the set A, and by proposition 4.3.10, $\mathit{Divides}(x, y)$ also represents the set A.

Section 5.3.1, Exercise 11, page 129:

Let $a \in \mathbb{N}$. Let $\top :\equiv (\forall x)[x = x]$. We will prove

$$N \vdash [(\forall x < \overline{a})\phi(x)] \rightarrow [\top \wedge \phi(\overline{0}) \wedge \ldots \wedge \phi(\overline{a-1})] \quad \text{(I)}$$

and

$$N \vdash [\top \wedge \phi(\overline{0}) \wedge \ldots \wedge \phi(\overline{a-1})] \rightarrow [(\forall x < \overline{a})\phi(x)] \quad \text{(II)}$$

and thus, $N \vdash [(\forall x < \overline{a})\phi(x)] \leftrightarrow [\top \wedge \phi(\overline{0}) \wedge \ldots \wedge \phi(\overline{a-1})]$ follows by (PC). (If $a = 0$, then the formulas in (I) and (II) are, respectively, $[(\forall x < \overline{a})\phi(x)] \rightarrow \top$ and $\top \rightarrow [(\forall x < \overline{a})\phi(x)]$. Note that $N \vdash \top$.)

First, we prove (I). Fix an arbitrary $a \in \mathbb{N}$. Recall that $(\forall x < \overline{a})\phi(x)$ is shorthand for $(\forall x)[x < \overline{a} \rightarrow \phi(x)]$. Thus, by (Q1)

$$N \vdash [(\forall x < \overline{a})\phi(x)] \rightarrow [\overline{b} < \overline{a} \rightarrow \phi(\overline{b})] \quad \text{(i)}$$

for any $b \in \mathbb{N}$. By Lemma 2.8.4(3), we have $N \vdash \overline{b} < \overline{a}$ for any b such that $b < a$. Thus, by Lemma 2.8.4(3), (i) and (PC), we have

$$N \vdash [(\forall x < \overline{a})\phi(x)] \rightarrow \phi(\overline{b}) \quad \text{(ii)}$$

for any $b < a$. By (ii) and (PC), we have

$$N \vdash [(\forall x < \overline{a})\phi(x)] \rightarrow [\top \wedge \phi(\overline{0}) \wedge \ldots \wedge \phi(\overline{a-1})] .$$

This shows that (I) holds. We turn to the proof of (II). By (E3) and other logical axioms, we have

$$\vdash \phi(\overline{b}) \rightarrow (x = \overline{b} \rightarrow \phi(x)) \quad \text{(iii)}$$

for any $b \in \mathbb{N}$. Next, note that $(A_0 \wedge A_1) \rightarrow ((B_0 \vee B_1) \rightarrow C)$ follows tautologically from $A_0 \rightarrow (B_0 \rightarrow C)$ and $A_1 \rightarrow (B_1 \rightarrow C)$. Hence, by (iii) and (PC), we have

$$\vdash [\top \wedge \phi(\overline{0}) \wedge \ldots \wedge \phi(\overline{a-1})] \rightarrow [(x = \overline{0} \vee \ldots \vee x = \overline{a-1}) \rightarrow \phi(x)]. \quad \text{(iv)}$$

By Rosser's Lemma, we have

$$N \vdash x < \overline{a} \rightarrow (x = \overline{0} \vee \ldots \vee x = \overline{a-1}). \quad \text{(v)}$$

By (iv), (v), and (PC), we have

$$N \vdash [\top \wedge \phi(\overline{0}) \wedge \ldots \wedge \phi(\overline{a-1})] \rightarrow [x < \overline{a} \rightarrow \phi(x)]. \quad \text{(vi)}$$

By (vi) and (QR), we have

$$N \vdash [\top \wedge \phi(\overline{0}) \wedge \ldots \wedge \phi(\overline{a-1})] \rightarrow (\forall x)[x < \overline{a} \rightarrow \phi(x)] \quad \text{(vii)}$$

and (vii) is nothing but (II) since $(\forall x < \overline{a})\phi(x)$ is shorthand for $(\forall x)[x < \overline{a} \rightarrow \phi(x)]$.

Section 5.3.1, Exercise 12, page 129:

Case: ϕ is of the form $(\alpha \vee \beta)$. We have $\phi(\underset{\sim}{x}, \underset{\sim}{y}, \underset{\sim}{z}) :\equiv \alpha(\underset{\sim}{x}, \underset{\sim}{y}) \vee \beta(\underset{\sim}{x}, \underset{\sim}{z})$ where $\underset{\sim}{x}$ are the free variables of ϕ with free occurrences in both α and β; $\underset{\sim}{y}$ are the free variables of ϕ with no free occurrences in β; $\underset{\sim}{z}$ are the free variables of ϕ with no free occurrences in α. By our induction hypothesis, we have

$$\mathfrak{N} \models \alpha(\underset{\sim}{t}, \underset{\sim}{s}) \quad \Rightarrow \quad N \vdash \alpha(\underset{\sim}{t}, \underset{\sim}{s}) \tag{i}$$

and

$$\mathfrak{N} \models \beta(\underset{\sim}{t}, \underset{\sim}{r}) \quad \Rightarrow \quad N \vdash \beta(\underset{\sim}{t}, \underset{\sim}{r}) \tag{ii}$$

for any variable-free terms $\underset{\sim}{t}, \underset{\sim}{s}, \underset{\sim}{r}$. Now, assume $\mathfrak{N} \models \alpha(\underset{\sim}{t}, \underset{\sim}{s}) \vee \beta(\underset{\sim}{t}, \underset{\sim}{r})$ for some variable-free terms $\underset{\sim}{t}, \underset{\sim}{s}, \underset{\sim}{r}$. Then, $\mathfrak{N} \models \alpha(\underset{\sim}{t}, \underset{\sim}{s})$ or $\mathfrak{N} \models \beta(\underset{\sim}{t}, \underset{\sim}{r})$.

First we assume that we have $\mathfrak{N} \models \alpha(\underset{\sim}{t}, \underset{\sim}{s})$. By (i), we have $N \vdash \alpha(\underset{\sim}{t}, \underset{\sim}{s})$. By (PC), we have $N \vdash \alpha(\underset{\sim}{t}, \underset{\sim}{s}) \vee \beta(\underset{\sim}{t}, \underset{\sim}{r})$.

Next assume that $\mathfrak{N} \not\models \alpha(\underset{\sim}{t}, \underset{\sim}{s})$. Then, $\mathfrak{N} \models \beta(\underset{\sim}{t}, \underset{\sim}{r})$. By (ii), we have $N \vdash \beta(\underset{\sim}{t}, \underset{\sim}{r})$. By (PC), we have $N \vdash \alpha(\underset{\sim}{t}, \underset{\sim}{s}) \vee \beta(\underset{\sim}{t}, \underset{\sim}{r})$.

Case: ϕ is of the form $(\alpha \wedge \beta)$. This case is similar to the case for the form $(\alpha \vee \beta)$.

Case: ϕ is of the form $(\exists y)\psi$. We have $\phi(\underset{\sim}{x}) :\equiv (\exists y)\psi(y, \underset{\sim}{x})$. Our induction hypothesis is

$$\mathfrak{N} \models \psi(s, \underset{\sim}{t}) \quad \Rightarrow \quad N \vdash \psi(s, \underset{\sim}{t})$$

for any variable-free terms $s, \underset{\sim}{t}$. Now, assume $\mathfrak{N} \models (\exists y)\psi(y, \underset{\sim}{t})$ for some variable-free terms $\underset{\sim}{t}$. Then there exists an $a \in \mathbb{N}$ such that $\mathfrak{N} \models \psi(\overline{a}, \underset{\sim}{t})$. As $\overline{a}$ is variable-free term, our induction hypothesis yields $N \vdash \psi(\overline{a}, \underset{\sim}{t})$. Furthermore, $\psi(\overline{a}, \underset{\sim}{t}) \rightarrow (\exists y)\psi(y, \underset{\sim}{t})$ is an instance of the axiom scheme (Q2). Thus, by (PC), we conclude that $N \vdash (\exists y)\psi(y, \underset{\sim}{t})$.

Case: ϕ is of the form $(\forall y < u)\psi$. We have $\phi(\underset{\sim}{x}) :\equiv (\forall y < u)\psi(y, \underset{\sim}{x})$. (The variables in the term u are included in the list $\underset{\sim}{x}$.) The induction hypothesis is

$$\mathfrak{N} \models \psi(s, \underset{\sim}{t}) \quad \Rightarrow \quad N \vdash \psi(s, \underset{\sim}{t})$$

for any variable-free terms $s, \underset{\sim}{t}$. Let $u(\underset{\sim}{t})$ denote u where the variables $\underset{\sim}{x}$, respectively, are replaced by the terms $\underset{\sim}{t}$. Note that $u(\underset{\sim}{t})$ is a variable-free term when $\underset{\sim}{t}$ are variable-free terms.

Assume $\mathfrak{N} \models (\forall y < u(\underset{\sim}{t}))\psi(y, \underset{\sim}{t})$ for some variable-free terms $\underset{\sim}{t}$. This entails that $\mathfrak{N} \models \psi(\overline{a}, \underset{\sim}{t})$ for every $a < u(\underset{\sim}{t})^{\mathfrak{N}}$. By our induction hypothesis, we have $N \vdash \psi(\overline{a}, \underset{\sim}{t})$ for every $a < u(\underset{\sim}{t})^{\mathfrak{N}}$. By (PC), we have

$$N \vdash \psi(\overline{0}, \underset{\sim}{t}) \ \wedge \ \psi(\overline{1}, \underset{\sim}{t}) \ \wedge \ \ldots \ \wedge \ \psi(\overline{u(\underset{\sim}{t})^{\mathfrak{N}} - 1}, \underset{\sim}{t}) \ .$$

By Corollary 4.3.8 and (PC), we have $N \vdash (\forall y < \overline{u(\underset{\sim}{t})^{\mathfrak{N}}}))\psi(y, \underset{\sim}{t})$.

We are not yet done. So far we have established that

$$N \vdash (\forall y < \overline{u(\underset{\sim}{t})^{\mathfrak{N}}}))\psi(y, \underset{\sim}{t}) \tag{iii}$$

but we should prove that $N \vdash (\forall y < u(\underset{\sim}{t}))\psi(y, \underset{\sim}{t})$. Now we need Lemma 4.3.6. This lemma yields

$$N \vdash u(\underset{\sim}{t}) = \overline{u(\underset{\sim}{t})^{\mathfrak{N}}} \ . \tag{iv}$$

Furthermore, we have

$$\vdash \ u(\underset{\sim}{t}) = \overline{u(\underset{\sim}{t})^{\mathfrak{N}}} \ \rightarrow \ [(\forall y < \overline{u(\underset{\sim}{t})^{\mathfrak{N}}}))\psi(y, \underset{\sim}{t}) \ \rightarrow \ (\forall y < u(\underset{\sim}{t})))\psi(y, \underset{\sim}{t})] \tag{v}$$

as the formula holds in every $\mathcal{L}_{NT}$-structure. (By the Completeness Theorem, the formula in (v) is derivable from the logical axioms Λ.) By (iii), (iv), (v) and (PC), we conclude that $N \vdash (\forall y < u(\underset{\sim}{t}))\psi(y, \underset{\sim}{t})$.

Section 5.4.1

Section 5.4.1, Exercise 3, page 133:

(a) First, suppose that A is semi-calculable, and suppose that the program P confirms that potential elements of A are, in fact, elements of A but P never halts when provided a non-element of A. We must show that A is listable by describing a computer program L that lists the elements of A. Here is what L does: First, it runs P on input 0 for one minute. If P halts and says that $0 \in A$, L then prints the number 0. If not, then L remembers the state of program P after that minute, and proceeds to run P on input 1 for one minute, again printing out 1 if P halts during that minute, or saving the state of the computer after that minute and then returning to run P on 0, picking up where the program had halted before. And we continue in this fashion, spending a minute on each computation in the following inputs in this order, picking up any interrupted computation at the point where it was halted:

$$0, 1, 0, 1, 2, 0, 1, 2, 3, 0, 1, 2, 3, 4, 0, 1, 2, 3, 4, 5, \ldots$$

This scheme will run P on each input for as long as needed for P to return 1 if that input is an element of A, so L will eventually print that number in the list it generates. If a number is not an element of A, program P won't halt, and thus L won't print that number. Thus L lists A, as needed.

For the converse, assume that A is listable and suppose that L prints the elements of A. Here's what program P will do on input k: Run the program L, and if L ever prints the number k, P answers 1.

That's it. You can check that the program P performs as needed to show that A is semi-calculable.

Section 5.4.1, Exercise 4, page 133:

To define the set B, just run the program L that lists A (from Exercise 3), but suppress the printing of any number that is less than or equal to the number that you have just printed. This will print an infinite (as A is infinite) increasing list of numbers. By the previous exercise, B is calculable.

Section 5.8.1

Section 5.8.1, Exercise 3, page 147:

The collection of terms is the closure of the collection of variables and constant symbols under the application of the function symbols.

Base Case. If t is the constant symbol 0 (consisting of a single symbol), then $\ulcorner 0 \urcorner = \langle 9 \rangle = 2^{10} = 1024 > 1$. The variable with the

smallest Gödel number is v_1, and if we look at $\ulcorner v_1 \urcorner$, we see $\ulcorner v_1 \urcorner = \langle 2 \rangle = 2^3 = 8$, and 8 is greater than the number of symbols in v_1.

Inductive Case. Suppose that the Lemma is true of the term t and we wish to show that it is true of the term St. If t has k symbols, then St has $k+1$ symbols, and

$$\ulcorner St \urcorner = \langle 11, \ulcorner t \urcorner \rangle = 2^{12} 3^{\ulcorner t \urcorner + 1} > 2^{12} 3^k > k+1,$$

as needed. The other inductive cases are handled in a similar fashion.

Section 5.12.1

Section 5.12.1, Exercise 5, page 166:

To start this, we have to change the given

$$\phi(x,y) \text{ is } \big[[\forall x P(x)] \to (Q(x,y) \to [\forall x P(x)]) \big]$$

to something that uses real quantifiers. It's pretty quick to see that ϕ is really

$$\phi(x,y) \text{ is } \big[\neg [\forall x P(x)] \vee (\neg Q(x,y) \vee [\forall x P(x)]) \big].$$

Then (assuming that P and Q are relation symbols in the language) a construction sequence for ϕ could be

$$\big(P(x), Q(x,y), \forall x P(x), \neg [\forall x P(x)], \neg Q(x,y), (\neg Q(x,y) \vee [\forall x P(x)]), \big[\neg [\forall x P(x)] \vee (\neg Q(x,y) \vee [\forall x P(x)]) \big] \big).$$

The prime components of ϕ are simply $\forall x P(x)$ and $Q(x,y)$. Since there are only two prime components, the list of all possible truth assignments is reasonably short: $(0,0), (0,1), (1,0), (1,1)$. It is easy to check with all four of these assignments that the truth of ϕ is 1 in each case. Which is a good thing.

Section 6.2.1

Section 6.2.1, Exercise 1, page 173:

We need to prove that

$$N \vdash \Big(y = \overline{\mathrm{R}(a)} \to Rf(\overline{a}, y) \Big).$$

By (E3) and other logical axioms we have

$$\vdash y = \overline{\mathrm{R}(a)} \;\to\; \big(Rf(\overline{a}, \overline{\mathrm{R}(a)}) \;\to\; Rf(\overline{a}, y) \big). \qquad \text{(i)}$$

By the first claim of the lemma we know that the formula Rf represents the function R. Hence

$$N \vdash Rf(\overline{a}, \overline{\mathrm{R}(a)}). \tag{ii}$$

By (i), (ii) and (PC), we have

$$N \vdash \left(y = \overline{\mathrm{R}(a)} \;\rightarrow\; Rf(\overline{a}, y)\right).$$

Section 6.2.1, Exercise 3, page 174:

First, let's work with $\psi(v_1)$ being $Formula(v_1)$. If we look at the sentence ϕ, then certainly $\mathfrak{N} \models Formula(\overline{\ulcorner \phi \urcorner})$, and since $Formula(\overline{\ulcorner \phi \urcorner})$ is a true Δ-sentence, $N \vdash Formula(\overline{\ulcorner \phi \urcorner})$. Since $N \vdash \big(\phi \leftrightarrow Formula(\overline{\ulcorner \phi \urcorner})\big)$, we see that $N \vdash \phi$.

Since $N \vdash \phi$ and N is consistent (it has a model), $N \nvdash \neg\phi$.

Now, if instead we use $\psi(v_1) = \neg Formula(v_1)$ to generate ϕ, it is still that case that $Formula(\overline{\ulcorner \phi \urcorner})$ is a true Δ-sentence and so $N \vdash Formula(\overline{\ulcorner \phi \urcorner})$. So in this case, $N \vdash \neg\phi$ and N does not prove ϕ.

Section 6.2.1, Exercise 4, page 174:

We have $N \vdash \big(\phi \leftrightarrow Even(\overline{\ulcorner \phi \urcorner})\big)$. As ϕ is a sentence, it is a true statement in the natural numbers that $\ulcorner \phi \urcorner$ is an even number, and as $Even(v_1)$ is a Δ-formula, $N \vdash Even(\overline{\ulcorner \phi \urcorner})$. Hence $N \vdash \phi$, and $N \nvdash \neg\phi$.

Section 6.3.1

Section 6.3.1, Exercise 1, page 181:

Let $B = \{\sigma \mid \sigma \text{ is a sentence and } A \vdash \sigma\}$. We must show that B is a theory. To that end, assume that $\hat{\sigma}$ is a sentence and that $B \vdash \hat{\sigma}$. We must prove that $\hat{\sigma} \in B$. So we must show that there is a deduction-from-A of the sentence $\hat{\sigma}$.

Let D be our given deduction-from-B of $\hat{\sigma}$. In that deduction, certain formulas are listed whose only justification is that they are elements of B. Make a list $\phi_1, \phi_2, \ldots, \phi_k$ of those formulas. Since each ϕ_i is an element of B, we know that there is a deduction-from-A of ϕ_i. Pick such a deduction and call it D_i. Now consider the deduction

$$D_1\, D_2\, \ldots D_k\, D.$$

This is a list of formulas and each of the formulas ϕ_i that shows up in the D part of the deduction can be justified as being just copied from earlier in the deduction. This means that this long deduction is a deduction-from-A, and as $\hat{\sigma}$ is the last line of D, this deduction proves that $A \vdash \hat{\sigma}$, and thus that B is a theory.

Section 6.3.1, Exercise 2, page 181:

This is similar to the previous exercise. Assume that $Th(\mathfrak{A}) \vdash \sigma$ for a sentence σ. We must prove that $\sigma \in Th(\mathfrak{A})$. In other words, we must show that $\mathfrak{A} \models \sigma$. But we know that $\mathfrak{A}$ is a model of $Th(\mathfrak{A})$, and so any sentence that $Th(\mathfrak{A})$ proves must be true in $\mathfrak{A}$ (that's Soundness). This means that $\mathfrak{A} \models \sigma$, as needed.

Section 6.3.1, Exercise 3, page 181: :

If the set of axioms A is strong enough to prove the axioms of N, then the answer is no. Any Σ-sentence that is true in the standard model is provable by N, and since $A \vdash N$, A would be strong enough to prove any true Σ-sentence.

Section 6.4.1

Section 6.4.1, Exercise 1, page 185:

First, assume that B is consistent and $A \subseteq B$. We must show that A is consistent. By way of contradiction, if A is inconsistent, then $A \vdash \bot$. But any element of A is also an element of B, so this means that $B \vdash \bot$, as well. This means that B is inconsistent, contrary to assumption.

Now, assume that B is ω-consistent. If we assume that A is ω-inconsistent, then we know the following:

$$A \vdash \exists x \phi(x)$$
$$A \vdash \neg\phi(\overline{n}) \text{ for each } n \in \mathbb{N}$$

And once again, as B extends A, B can prove the above formulas as well. Hence B would be ω-inconsistent, contrary to assumption. Thus if B is ω-consistent, so is A.

Section 6.4.1, Exercise 4, page 185:

Two ways to attack this. First, the obvious way:

Suppose that θ were true. If $\mathfrak{N} \models \theta$, then by choice of θ we would know that $\mathfrak{N} \models Thm_N\left(\overline{\ulcorner \neg\theta \urcorner}\right)$. This tells us that $N \vdash \neg\theta$, and so $\mathfrak{N} \models \neg\theta$, a contradiction. So θ cannot be true.

This tells us that θ can't be provable, either, as N only proves things that are true (in the standard model).

What about refutability? Suppose that θ were refutable. Then we would know that $N \vdash \neg\theta$, and $Thm_N\left(\overline{\ulcorner \neg\theta \urcorner}\right)$ would be a true Σ-sentence. Thus $N \vdash Thm_N\left(\overline{\ulcorner \neg\theta \urcorner}\right)$ and by choice of θ, we would have $N \vdash \theta$, making N inconsistent, a contradiction. Hence θ is not refutable.

Here's the quick way: Look at $\neg\theta$. By choice of θ, we have

$$N \vdash \left[\theta \leftrightarrow Thm_N\left(\overline{\ulcorner\neg\theta\urcorner}\right)\right],$$

so

$$N \vdash \left[\neg\theta \leftrightarrow \neg Thm_N\left(\overline{\ulcorner\neg\theta\urcorner}\right)\right].$$

This means that $\neg\theta$ asserts its own unprovability. By the argument for Gödel's First Incompleteness Theorem, we know that $\neg\theta$ is true and unprovable. This makes θ false and not refutable.

Section 6.5.1

Section 6.5.1, Exercise 2, page 187:
Suppose that $\phi(x)$ named both 17 and 42 with respect to some Q that extends N. Then, as (after a bit of work) $Q \vdash \phi(\overline{17})$ and $Q \vdash \phi(\overline{42})$, we'd have $Q \vdash \overline{17} = \overline{42}$, which would contradict, for example, Lemma 2.8.4, as Q proves all the axioms of N.

Section 6.5.1, Exercise 4, page 187:
This is pretty automatic. Since n is the only element of $\{n\}$, we need to show that $N \vdash \phi(\overline{n})$ and, for every $k \neq n$, $N \vdash \neg\phi(\overline{k})$. Both follow almost instantly from Lemma 2.8.4.

Section 7.3.1

Section 7.3.1, Exercise 1, page 212:
(a) We have $\mathrm{sg}(x) = 1 \dot{-} (1 \dot{-} x)$. Thus, sg is primitive recursive by Lemma 7.3.1 and Lemma 7.3.3.

Section 7.3.1, Exercise 3, page 213:
(a) The predicate $D(x, y)$ holds if, and only if

$$(\exists z \leq y)\left[x \cdot z = y \ \wedge \ z > 1\right].$$

Hence, by our lemmas, D is a primitive recursive predicate.

(b) We say that t encodes the sequence $x_0, \ldots x_n$ when $t = p(1)^{x_0} \cdot \ldots \cdot p(n+1)^{x_n}$. When t encodes $x_0, \ldots x_n$, then we have $\pi_{i+1}(t) = x_i$ for $i = 0, \ldots, n$. Moreover, when t encodes $x_0, \ldots x_n$, we have $t \leq p(x_1 + \ldots + x_n)^{p(x_1+\ldots+x_n)}$

Let D be the predicate from (a), and let t encode $x_0, \ldots x_n$. Then, the predicate

$$(\forall i \leq n)[D(\pi_{i+1}(t), x)]$$

states that x_i is a proper divisor of x (for $i = 0, \ldots, n$), and the predicate

$$(\sum_{i \leq n} \pi_{i+1}(t)) = x$$

states that $x_0 + x_1 + \ldots + x_n = x$.

Let $P(x)$ be the predicate

$$(\exists t \leq p(x)^{p(x)})(\exists n \leq t) \left[(\sum_{i \leq n} \pi_{i+1}(t)) = x \ \wedge \ (\forall i \leq n)[D(\pi_{i+1}(t), x)] \right] .$$

By our lemmas, P is a primitive recursive predicate. Moreover, $P(x)$ states that x is a perfect number.

We are asked to prove that the set of all perfect numbers is primitive recursive. We have a primitive recursive predicate $P(x)$ that holds iff x is a perfect number. By definition, a predicate is primitive recursive if the characteristic function for the predicate is primitive recursive. The characteristic function χ_P (for the predicate P) is also the the characteristic function for the set of all perfect numbers. A predicate is nothing but a set.

Section 7.3.1, Exercise 4, page 213:

Let a_n denote the n^{th} Fibonacci number. (Recall that $p(1) = 2$, $p(2) = 3$, $p(3) = 5$, and so on, whereas $p(0) = 1$) Let f_0 be the function given by

$$f_0(n) \ = \ p(1)^{a_0} \cdot p(2)^{a_1} \cdot p(3)^{a_2} \cdot \ldots \cdot p(n+1)^{a_n} .$$

Then, $f_0(n)$ encodes the sequence $a_0, a_1, \ldots, a_n$ and $\pi_{n+1}(f_0(n)) = a_n$.

We argue that f_0 is a primitive recursive function: Observe that $a_n \leq 2^n$. It follows that

$$f_0(n) \ < \ \left(p(n+1)^{2^{n+1}}\right)^{n+1} \ = \ p(n+1)^{(n+1)2^{n+1}} .$$

Thus, we have

$$\begin{aligned} f_0(n) \ = \ & (\mu t \leq p(n+1)^{(n+1)2^{n+1}}) \left[\pi_1(t) = 1 \ \wedge \ \pi_2(t) = 1 \ \wedge \right. \\ & \left. (\forall i \leq n+1)[i > 2 \ \rightarrow \ \pi_{i-2}(t) + \pi_{i-1}(t) = \pi_i(t)] \right] \quad (*) \end{aligned}$$

Now, all the functions and relations appearing in (*) are primitive recursive. The primitive recursive relations are closed under propositional connectives and bounded quantification. The primitive recursive functions are closed under bounded minimalization. Hence, we conclude that f_0 is a primitive recursive function.

Let $f(n) = \pi_{n+1}(f_0(n))$. Then, f is a primitive recursive function. We have $f(n) = a_n$.

Section 7.3.1, Exercise 5, page 213:

We define h by $h(x,y,z) = \mathcal{S}(\mathcal{I}_3^3(x,y,z))$, and we define f by

$$f(x,0) = \mathcal{I}_1^1(x) \quad \text{and} \quad f(x,y+1) = h(x,y,f(x,y)) \; .$$

Then we have $f(x,y) = x + y$ (see Section 7.2, page 200). We define the function f_1 by

$$f_1(x) = \mathcal{O} \tag{*}$$

Note that (*) indeed is of the form

$$f(x_1,\ldots,x_n) = h(g_1(x_1,\ldots,x_n),\ldots,g_m(x_1,\ldots,x_n))$$

when $n = 1$ and $m = 0$. Furthermore, we define f_2 by

$$f_2(x,z,y) = f(\mathcal{I}_1^3(x,z,y),\mathcal{I}_3^3(x,z,y)) \; .$$

Then, we have $f(x,z,y) = x + y$. Finally, we define g by

$$g(x,0) = f_1(x) \quad \text{and} \quad g(x,y+1) = f_2(x,y,g(x,y)) \; .$$

Now we have $g(x,y) = x \cdot y$.

Section 7.3.1, Exercise 8, page 214:

(a) Let f be a function of rank 0. We prove that there exists $i,k \in \mathbb{N}$ such that

$$f(x_1,\ldots,x_n) \;\leq\; x_i + k \tag{*}$$

by induction on the structure of f. (Any function of rank 0 that is not an initial function can be defined by the scheme of composition.)

It is trivial that (*) holds when f is one of the initial functions. Assume $f(\underset{\sim}{x}) = h(g_1(\underset{\sim}{x}),\ldots,g_m(\underset{\sim}{x}))$ where $\underset{\sim}{x} = x_1,\ldots,x_n$. The induction hypothesis yields i_j and k_j such that

$$g_j(x_1,\ldots,x_n) \;\leq\; x_{i_j} + k_j \qquad \text{(for } j = 1,\ldots,m.)$$

The induction hypothesis also yield j and ℓ such that

$$h(x_1,\ldots,x_m) \;\leq\; x_j + \ell \; .$$

Thus, we have

$$\begin{aligned} f(x_1,\ldots,x_n) \;=\; & h(g_1(x_1,\ldots,x_n),\ldots,g_m(x_1,\ldots,x_n)) \;\leq \\ & g_j(x_1,\ldots,x_n) + \ell \;\leq\; x_{i_j} + k_j + \ell \end{aligned}$$

and (*) holds when i is i_j and k is $k_j + \ell$. This proves (*).

(c) We prove that the multiplication function is not of rank 1. Assume it is. By (b), we have a fixed $k \in \mathbb{N}$ such that

$$x \cdot y \ < \ k(\max(x, y) + 1)$$

holds for any $x, y \in \mathbb{N}$. But this is not true. Let $x = y = k+1$. Then,

$$x \cdot y \ = \ (k+1)^2 \ = \ k^2 + 2k + 1$$

whereas

$$k(\max(x, y) + 1) \ = \ k(\max(k+1, k+1) + 1) \ = \ k^2 + 2k \ .$$

The function 2^x is a function of rank 2. The function $x + y$ is of rank 1. Thus, the function $2x$ is also of rank 1 as $2x = x + x$. Thus, 2^x is a function of rank 2 as $2^0 = 1$ and $2^{x+1} = 2 \cdot 2^x$.

Section 7.3.1, Exercise 9, page 214:

(b) We prove that the function A_n is primitive recursive by induction on n. It is obvious that A_0 is primitive recursive as $A_0(x) = x + 2$. Assume that A_n is primitive recursive. Observe that $A_{n+1}(0) = 2$ and that $A_{n+1}(x + 1) = A_n(A_{n+1}(x))$. Hence, A_{n+1} can be defined from A_n by primitive recursive definition schemes, and we conclude that A_{n+1} is a primitive recursive function.

(c) Let f be a function of rank n. We prove that there exists k such that

$$f(x_1, \ldots, x_\ell) \ \leq \ A_n^k(\max(x_1, \ldots, x_\ell)) \qquad (*)$$

by induction on the structure of f. We will use that $x \leq A_n(x)$, that $A_n(x) \leq A_{n+1}(x)$, and that $A_n(x) \leq A_n(x + 1)$ (see (a)).

Assume f is one of initial functions. Then f is of rank 0, and it is obvious that (*) holds as $A_0(x) = x + 2$.

Assume $f(\underset{\sim}{x}) = h(g_1(\underset{\sim}{x}), \ldots, g_m(\underset{\sim}{x}))$. Furthermore, assume that f is of rank n. Then, $h, g_1, \ldots, g_m$ are of rank less than or equal to n. By our induction hypothesis we have $k_0, k_1, \ldots, k_m \in \mathbb{N}$ such that

$$h(x_1, \ldots, x_m) \ \leq \ A_n^{k_0}(\max(x_1, \ldots, x_m)) \qquad (1)$$

and

$$g_i(\underset{\sim}{x}) \ \leq \ A_n^{k_i}(\max(\underset{\sim}{x})) \qquad (2)$$

for $i = 1, \ldots, m$. Thus

$$\begin{aligned} f(\underset{\sim}{x}) \quad &= \quad h(g_1(\underset{\sim}{x}), \ldots, g_m(\underset{\sim}{x})) \\ &\leq \quad A_n^{k_0}(\max(g_1(\underset{\sim}{x}), \ldots, g_m(\underset{\sim}{x}))) && (1) \\ &\leq \quad A_n^{k_0}(\max(A_n^{k_1}(\max(\underset{\sim}{x})), \ldots, A_n^{k_m}(\max(\underset{\sim}{x})))) && (2) \\ &\leq \quad A_n^{k_0 + \max(k_1, \ldots, k_m)}(\max(\underset{\sim}{x})). \end{aligned}$$

We see that (*) holds when $k = k_0 + \max(k_1, \ldots, k_m)$.

Assume $f(\underset{\sim}{x}, 0) = g(\underset{\sim}{x})$ and $f(\underset{\sim}{x}, y+1) = h(\underset{\sim}{x}, y, f(\underset{\sim}{x}, y))$. Then, we have $n \in \mathbb{N}$ such that f is of rank $n+1$ and $h, g_1, \ldots, g_m$ are of rank less than or equal to n. By our induction hypothesis we have $k_0, k_1 \in \mathbb{N}$ such that

$$g(\underset{\sim}{x}) \;\leq\; A_n^{k_0}(\max(\underset{\sim}{x})) \tag{3}$$

and

$$h(\underset{\sim}{x}, y, z) \;\leq\; A_n^{k_1}(\max(\underset{\sim}{x}, y, z)) \,. \tag{4}$$

We will prove by (a secondary) induction on y that

$$f(\underset{\sim}{x}, y) \;\leq\; A_n^{k_1 y + k_0}(\max(\underset{\sim}{x}, y)) \,. \tag{**}$$

It follows straightforwardly from (3) that (**) holds when $y = 0$. Furthermore, we have

$$\begin{array}{lcll} f(\underset{\sim}{x}, y+1) & = & h(\underset{\sim}{x}, y, f(\underset{\sim}{x}, y)) & \\ & \leq & A_n^{k_1}(\max(\underset{\sim}{x}, y, f(\underset{\sim}{x}, y+1))) & (4) \\ & \leq & A_n^{k_1}(\max(\underset{\sim}{x}, y, A_n^{k_1 y + k_0}(\max(\underset{\sim}{x}, y)))) & \text{ind. hyp.} \\ & = & A_n^{k_1(y+1)+k_0}(\max(\underset{\sim}{x}, y)) \,. & \end{array}$$

This proves (**).

Now we have

$$\begin{array}{lcll} f(\underset{\sim}{x}, y) & \leq & A_n^{k_1 y + k_0}(\max(\underset{\sim}{x}, y)) & (**) \\ & \leq & A_n^{k_1 y + k_0}(A_{n+1}(\max(\underset{\sim}{x}, y))) & \\ & \leq & A_{n+1}(k_1 y + k_0 + \max(\underset{\sim}{x}, y)) & \text{def. of } A_{n+1} \\ & \leq & A_{n+1}(A_1^{k_1+k_0+1}(\max(\underset{\sim}{x}, y)) & \text{see (a)} \\ & \leq & A_{n+1}(A_{n+1}^{k_1+k_0+1}(\max(\underset{\sim}{x}, y))) & \\ & = & A_{n+1}^{k_1+k_0+2}(\max(\underset{\sim}{x}, y)) & \end{array}$$

We conclude that (*) holds when $k = k_1 + k_0 + 2$.

(d) Assume that A is primitive recursive. Let $d(x) = d_0(x, x)$ where $d_0(x, 0) = x$ and $d_0(x, y+1) = A(x, d_0(x, y))$. Then, d is primitive recursive since A is primitive recursive. Moreover, $d(x) = A_x^x(x)$. But by (c), we have fixed $k, n \in \mathbb{N}$ such that $d(x) \leq A_n^k(x)$, and a contradiction emerges: for any x strictly greater than $\max(n, k)$ we have

$$d(x) \;\leq\; A_n^k(x) \;<\; A_x^x(x) \;=\; d(x) \,.$$

Section 7.4.1

Section 7.4.1, Exercise 1, page 223:

According to Definition 7.4.1

- $\langle 0 \rangle$ is an index for $\mathcal{S}$
- $\langle 1, 3, 3 \rangle$ is an index for $\mathcal{I}_3^3$
- $\langle\ 3\ ,\ 3\ ,\ \langle 1, 3, 3 \rangle\ ,\ \langle 0 \rangle\ \rangle$ is an index for h
- $\langle\ 4\ ,\ 2\ ,\ \langle 1, 1, 1 \rangle\ ,\ \langle 3, 3, \langle 1, 3, 3 \rangle, \langle 0 \rangle \rangle\ \rangle$ is an index for f.

The exact value of $\langle 4, 2, \langle 1, 1, 1 \rangle, \langle 3, 3, \langle 1, 3, 3 \rangle, \langle 0 \rangle \rangle \rangle$ depends on the convention for coding sequences of numbers, and

$$\langle 4, 2, \langle 1, 1, 1 \rangle, \langle 3, 3, \langle 1, 3, 3 \rangle, \langle 0 \rangle \rangle \rangle$$

will be a very long number in decimal notation when our coding conventions are based on prime factorization.

Section 7.4.1, Exercise 2, page 223:

Let $f(x) = \mathcal{S}(\mathcal{S}(x))$. Then f is defined by the scheme of composition. By Clause (1) and Clause (4) of Definition 7.4.1, the number $\langle 3, 1, \langle 0 \rangle, \langle 0 \rangle \rangle$ is a computable index for f.

Let $f_1(x) = f(\mathcal{I}_1^1(x))$, and let $f_2(x) = \mathcal{I}_1^1(f(x))$. Both f_1 and f_2 are defined by the scheme of composition. Moreover, $f(x) = f_1(x) = f_2(x) = x + 2$. The numbers

$$\langle\ 3\ ,\ 1\ ,\ \langle 1, 1, 1 \rangle\ ,\ \langle 3, 1, \langle 0 \rangle, \langle 0 \rangle \rangle\ \rangle \text{ and } \langle\ 3\ ,\ 1\ ,\ \langle 3, 1, \langle 0 \rangle, \langle 0 \rangle \rangle\ ,\ \langle 1, 1, 1 \rangle\ \rangle$$

are indices for, respectively, f_1 and f_2.

Section 7.4.1, Exercise 3, page 223:

Assume that f is defined from the computable functions g_1, g_2 and h by the scheme. We will prove that f is a computable function.

Let e_1 and e_2 be a computable indices for, respectively, g_1 and g_2, let $\underset{\sim}{x} = x_1, \ldots, x_n$, and let $\mathcal{U}$ and $\mathcal{T}_n$ be the function and the predicate given by Kleene's Normal Form Theorem. Let $P(\underset{\sim}{x}, t)$ be the predicate

$$(h(\underset{\sim}{x}) = 0\ \wedge\ \mathcal{T}_n(e_1, \underset{\sim}{x}, t))\ \ \vee\ \ (h(\underset{\sim}{x}) > 0\ \wedge\ \mathcal{T}_n(e_2, \underset{\sim}{x}, t))\ .$$

The Kleene T-predicate $\mathcal{T}_n$ is primitive recursive and, by the lemmas in Section 7.3, the computable predicate $P(\underset{\sim}{x}, t)$ is defined if, and only if, $h(\underset{\sim}{x})$ is defined.

Let $f_0(\underset{\sim}{x}) = (\mu t)[P(\underset{\sim}{x}, t)]$. Then, f_0 is a computable function since the class of computable functions is closed under minimalization. By Kleene's Normal Form Theorem, we have $f(\underset{\sim}{x}) = \mathcal{U}(f_0(\underset{\sim}{x}))$. Now, $\mathcal{U}$ is a primitive recursive, and thus a computable, function. The class of computable functions is closed under composition. This proves that f is a computable function.

Section 7.4.1, Exercise 5, page 223:

The number $\langle 3, 1, \langle 0\rangle, \langle 0\rangle\rangle$ is a computable index for A_0 as $A_0(x) = x + 2$. (See Exercise 2.) Moreover, $A_{n+1}(0) = 2$ and $A_{n+1}(y+1) = A_n(A_{n+1}(y+1))$. Thus, it easy to see that we have primitive recursive function f such that $f(n)$ is an index for A_n. By Kleene's Normal Form Theorem, we have $A(y, x) = A_y(x) = \mathcal{U}((\mu t)[\mathcal{T}_1(f(y), x, t)])$. The predicate $\mathcal{T}_1$ is computable. So are the functions $\mathcal{U}$ and f. Furthermore, computable functions are closed under composition and minimalization. Hence, we conclude that A is a computable function.

Section 7.4.1, Exercise 6, page 224:

We apply the scheme of composition

$$f(x_1, \ldots, x_n) = h(g_1(x_1, \ldots, x_n), \ldots, g_m(x_1, \ldots, x_n)) \qquad (*)$$

with $k = n$ and $m = 0$ and define c_0^k by

$$c_0^k(x_1, \ldots x_k) = \mathcal{O} \, .$$

By Clause (3) and (4) of Definition 7.4.1, $\langle 3, k, \langle 2\rangle\rangle$ is an index for c_0^k.

Furthermore, apply (*) with $k = n$ and $m = 1$ and define c_{i+1}^k by

$$c_{i+1}^k(x_1, \ldots x_k) = \mathcal{S}(c_i^k(x_1, \ldots x_k)) \, .$$

By Clause (1) and (4) of Definition 7.4.1, $\langle 3, k, e, \langle 0\rangle\rangle$ is an index for c_{i+1}^k when e is an index for c_i^k.

For $k \in \mathbb{N}$, let $q_k(0) = \langle 3, k, \langle 2\rangle\rangle$ and $q_k(y+1) = \langle 3, k, q_k(y), \langle 0\rangle\rangle$. Then, q_k is primitive recursive by the lemmas in Section 7.3, and $q_k(x)$ yields an index for c_x^k.

Section 7.4.1, Exercise 8, page 224:

Let e be a computable index for f (so e is a fixed number). Let $\imath(y) = S_1^1(e, y)$ where S_1^1 is the primitive recursive function given by the S-m-n Theorem. Then, $\imath$ is primitive recursive by the lemmas in Section 7.3. Furthermore, we have

$$\{\imath(y)\}(x) \;=\; \{S_1^1(e, y)\}(x) \;=\; \{e\}(y, x) \;=\; f(y, x) \, .$$

Section 7.4.1, Exercise 9, page 224:

(b) Let f be a computable function, and let S_1^1 be the primitive recursive function given by the S-m-n Theorem. Furthermore, let $g(y, x) = f(S_1^1(y, y), x)$. Now, g is obviously a computable function. Thus g has an index a. We have

$$\begin{aligned}
\{S_1^1(a, y)\}(x) &= \{a\}(y, x) && \text{(by the S-m-n Theorem)} \\
&= g(y, x) && \text{(as } a \text{ is an index for } g\text{)} \\
&= f(S_1^1(y, y), x) \, . && \text{(by the def. of } g\text{)}
\end{aligned}$$

The constant a depends on the definition of the function f, but the equation

$$\{S_1^1(a,y)\}(x) \;=\; f(S_1^1(y,y),x)$$

holds for any value of y, and in particular, we have

$$\{S_1^1(a,a)\}(x) \;=\; f(S_1^1(a,a),x)\;.$$

Since S_1^1 is a primitive recursive function, there exists $e \in \mathbb{N}$ such that $e = S_1^1(a,a)$. For this e, we have $\{e\}(x) = f(e,x)$.

Section 7.4.1, Exercise 10, page 224:
We have $\mathcal{I}_1^2(y,x) = y$. By the Second Recursion Theorem, we have e such that $\{e\}^1(x) = \mathcal{I}_1^2(e,x) = e$. Since $u(y,x) = \{y\}^1(x)$, we have $u(e,x) = e$.

Section 7.5.1

Section 7.5.1, Exercise 3, page 233:
(a) Let $P(t)$ be the predicate

$$\begin{aligned}&\mathrm{code}(t) \;\wedge\; |t| \geq 2 \;\wedge\; (t)_1 = 1 \;\wedge\; (t)_2 = 1 \;\wedge\\ &\qquad\qquad \forall_{i\leq|t|}\big[\, i \geq 3 \;\rightarrow\; (t)_i = (t)_{i\dot{-}1} + (t)_{i\dot{-}2}\,\big]\;.\end{aligned}$$

(b) Let $P(t)$ be the predicate

$$\begin{aligned}&\mathrm{code}(t) \;\wedge\; |t| \geq 2 \;\wedge\; (t)_1 = 1 \;\wedge\; (t)_2 = 1 \;\wedge\\ &\forall_{i\leq|t|}\big[\, i \geq 3 \;\rightarrow\; \exists_{a,b<i}[a \neq b \wedge a \geq 1 \wedge b \geq 1 \wedge (t)_i = (t)_a + (t)_b\,\big]\;.\end{aligned}$$

(c) Let $P(t)$ be the predicate

$$\begin{aligned}&\mathrm{code}(t) \;\wedge\; |t| \geq 2 \;\wedge\; (t)_1 = 1 \;\wedge\; (t)_2 = 1 \;\wedge\; \forall_{i\leq|t|}\big[\, i \geq 3 \;\rightarrow\\ &\qquad \exists_{v\leq t}\big\{\, |v| \geq 2 \;\wedge\; \forall_{a,b<|v|}[a \neq b \rightarrow (v)_{a+1} \neq (v)_{b+1}] \wedge\\ &\qquad\quad \forall_{a<|v|}[0 < (v)_{a+1} \wedge (v)_{a+1} < i] \;\wedge\; (t)_i = \sum_{j<|v|} (t)_{(v)_{j+1}}\,\big\}\,\big]\;.\end{aligned}$$

Section 7.5.1, Exercise 4, page 234:
(a) Let $P(t,n)$ be the predicate

$$\mathrm{code}(t) \;\wedge\; |t| = n{+}1 \;\wedge\; (t)_1 = 1 \;\wedge\; \forall_{i<|t|}\big[\, i > 0 \;\rightarrow\; (t)_{i+1} = \sum_{j<i}(t)_{j+1}\,\big]\;.$$

Then, $P(t,n)$ holds iff t encodes $c_0,\ldots,c_n$. Moreover, P is a primitive recursive predicate by the lemmas in Section 7.3.

(b) Let $f(n) = (\,(\mu t)[P(t,n)]\,)_{n+1}$. Then, $f(n) = c_n$. The function f is computable as the class of computable functions is closed under minimalization.

(c) Take any primitive recursive function g such that $\langle c_0, \ldots, c_n \rangle \leq g(n)$. Let

$$f(n) \;=\; (\,(\mu t \leq g(n))[\,P(t,n)\,]\,)_{n+1}\;.$$

Then, $f(n) = c_n$, and f is a primitive recursive function as the class of primitive recursive functions is closed under bounded minimalization.

Section 7.5.1, Exercise 5, page 234:

(b) Let $f(y) = (\mu t)[\mathcal{T}_1((y)_1, (y)_2, t)]$

(c) Assume that there exists a *total* computable function f such that

$$\{e\}^1(x) = m \;\;\Rightarrow\;\; m < f(\langle e, x \rangle)\;. \qquad (*)$$

We prove that the function d given by

$$d(x) = \begin{cases} \{x\}^1(x) + 1 & \text{if } \{x\}^1(x) \text{ is defined} \\ 0 & \text{if } \{x\}^1(x) \text{ is undefined.} \end{cases}$$

is computable. This contradicts Corollary 7.4.7.

Now, $\{e\}^1(x)$ is defined ($\{e\}^1(x) = m$ for some $m \in \mathbb{N}$) if, and only if, there exists t such that $\mathcal{T}_1(e, x, t)$ holds.

Let $f_e(x) = (\mu t)[\mathcal{T}_1, (e, x, t)]$. By the S-m-n Theorem, there exists a primitive recursive function ρ such that $\rho(e)$ is an index for f_e. Moreover, $f_e(x)$ is defined iff $\{e\}^1(x)$ is defined iff there exists t such that $\mathcal{T}_1(e, x, t)$ holds. By (*), we have a total computable function f such that

$$f_e(x) = m \;\;\Rightarrow\;\; m < f(\langle \rho(e), x \rangle)\;.$$

Then, $f_e(x)$ is defined if, and only if

$$(\exists t \leq f(\langle \rho(e), x \rangle))[\mathcal{T}_1(e, x, t)]\;. \qquad (**)$$

It is easy to see that (**) is a computable predicate. But then d is a computable function since $\{x\}^1(x)$ is defined if, and only if, $(\exists t \leq f(\langle \rho(x), x \rangle))[\mathcal{T}_1(x, x, t)]$ holds.

Section 7.6.1

Section 7.6.1, Exercise 4, page 243:

(a) Assume $A \neq \emptyset$. Let χ_A be the characteristic function of A, and let a be the least element of A. Let $f(0) = a$ and let

$$f(y+1) = \begin{cases} y+1 & \text{if } \chi_A(y+1) = 0 \text{ (i.e. if } y+1 \in A) \\ f(y) & \text{otherwise.} \end{cases}$$

Now, f is computable as the function χ_A is computable. Moreover, f is a total and non-decreasing function whose range $\text{rng}(f)$ equals A.

(b) The implication holds when A is finite since any finite set is computable. So, assume A is infinite and let $A = \text{rng}(f)$ where f is a total computable non-decreasing function. Let $f_0(x) = (\mu i)[f(i) \geq x]$. Then, f_0 is a total computable function. Let

$$g(x) = \begin{cases} 0 & \text{if } f(f_0(x)) = x \\ 1 & \text{otherwise.} \end{cases}$$

Now, g is computable, and g is the characteristic function for the set A. Hence, A is computable.

Section 7.6.1, Exercise 5, page 243:

Let χ_A and χ_B be the characteristic functions for, respectively, A and B. Then,

- $1 \dot{-} \chi_A(x)$ is the characteristic function for $\overline{A}$
- $\chi_A(x) \cdot \chi_B(x)$ is the characteristic function for $A \cap B$
- $1 \dot{-} (1 \dot{-} (\chi_A(x) + \chi_B(x)))$ is the characteristic function for $A \cup B$.

We have $A \cup B = \overline{\overline{A} \cap \overline{B}}$ and $A \cap B = \overline{\overline{A} \cup \overline{B}}$. Thus, if a class of sets is closed under complement and intersection, the class is also closed under union; and if a class of sets is closed under complement and union, the class is also closed under intersection.

Section 7.6.1, Exercise 6, page 243:

Let A and B be semi-computable sets.

By the definition of a semi-computable set we have computable functions g_A and g_B such that $A = \text{dom}(f_A)$ and $B = \text{dom}(f_B)$. Let h be any binary total computable function, and let $f(x) = h(f_A(x), f_B(x))$. Then, f is a computable function such that $A \cap B = \text{dom}(f)$. Hence, $A \cap B$ is a semi-computable set.

If A or B are empty, it is trivial that $A \cup B$ is a semi-computable set. So, assume that neither A nor B are finite. By Theorem 7.6.5, we have primitive recursive functions f_A and f_B such that $A = \text{rng}(f_A)$ and $B = \text{rng}(f_B)$. The function $\lfloor \frac{x}{2} \rfloor$ (x divided by 2 rounded down) is primitive recursive. Let

$$g(x) = \begin{cases} f_A(\lfloor \frac{x}{2} \rfloor) & \text{if } x \text{ is even} \\ f_B(\lfloor \frac{x}{2} \rfloor) & \text{otherwise.} \end{cases}$$

Then, g is a primitive recursive function such that $A \cup B = \text{rng}(g)$. By Theorem 7.6.5, $A \cup B$ is a semi-computable set.

Section 7.6.1, Exercise 8, page 243:

Assume that A and $\overline{A}$ are semi-computable sets. We prove that A is a computable set.

Let e and d be computable indices for, respectively, A and $\overline{A}$. Let

$$f(x) \quad = \quad (\mu t)[\,\mathcal{T}(e,x,t) \;\vee\; \mathcal{T}(d,x,t)\,]\;.$$

Then, f is a total computable function. Let

$$\chi(x) = \begin{cases} 0 & \text{if } \mathcal{T}(e,x,f(x)) \\ 1 & \text{otherwise.} \end{cases}$$

Then, χ is a total computable function. Moreover, χ is the characteristic function for the set A. Hence, A is a computable set.

Section 7.6.1, Exercise 9, page 243:

For the sake of a contradiction, assume that $\mathcal{K}$ is a computable set, and let $\chi_{\mathcal{K}}$ be the characteristic functions of $\mathcal{K}$. Furthermore, let

$$d(x) = \begin{cases} \{x\}^1(x)+1 & \text{if } \chi_{\mathcal{K}}(x) = 0 \\ 0 & \text{if } \chi_{\mathcal{K}}(x) = 1. \end{cases}$$

Then d is a computable function.

By our definitions, we have

$$\chi_{\mathcal{K}}(x) = 0 \quad \Leftrightarrow \quad x \in \mathcal{K} \quad \Leftrightarrow \quad x \in \mathcal{W}_x \quad \Leftrightarrow \quad x \in \mathrm{dom}(\{x\}^1)\;.$$

Hence, we have $\chi_{\mathcal{K}}(x) = 0$ if, and only if, $\{x\}^1(x)$ is defined. (Since $\chi_{\mathcal{K}}$ is a 0-1 valued function, we also have $\chi_{\mathcal{K}}(x) = 1$ if, and only if, $\{x\}^1(x)$ is undefined.) Thus, d is a computable function such that

$$d(x) = \begin{cases} \{x\}^1(x)+1 & \text{if } \{x\}^1(x) \text{ is defined} \\ 0 & \text{if } \{x\}^1(x) \text{ is undefined.} \end{cases}$$

Now we have contradiction as Corollary 7.4.7 states that d is not a computable function.

Section 7.6.1, Exercise 10, page 244:

(c) Let χ_A be the characteristic function of A. Then, χ_A is a total computable function. Let b_0 be a number in B, and let b_1 be a number that is not in B. Such b_0 and b_1 exist since B is a nontrivial set. Let

$$f(x) = \begin{cases} b_0 & \text{if } \chi_A(x) = 0 \\ b_1 & \text{otherwise.} \end{cases}$$

Then, f is a computable function. Moreover, we have $x \in A$ iff $f(x) \in B$.

(d) Let A be a computable set, and assume for the sake of a contradiction that $\mathcal{K} \leq_m A$. Then we have a total computable function f such that $x \in \mathcal{K}$ iff $f(x) \in A$. Let χ_A be the characteristic function of A, and let $\chi(x) = \chi_A(f(x))$. Then χ is a total computable function. Moreover, χ is the characteristic function of $\mathcal{K}$. Thus $\mathcal{K}$ is a computable set. This contradicts Theorem 7.6.12.

(e) Let f be a total computable function such that $x \in A$ iff $f(x) \in \mathcal{K}$. Let g be a computable function such that $\mathcal{K} = \mathrm{dom}(g)$. Such a g exists since $\mathcal{K}$ is semi-computable. Let $h(x) = g(f(x))$. Then h is a computable function such that $A = \mathrm{dom}(h)$. Thus, A is semi-computable.

(f) Assume that the set A is computable and nontrivial. Then, complement set $\overline{A}$ is also nontrivial. Hence, we have $A \leq_m \overline{A}$ by (c).

Assume that $A \leq_m \overline{A}$ and that A is a nontrivial semi-computable set. We will prove that A is computable. Since $A \leq_m \overline{A}$, we have a total computable function f such that

$$x \in A \quad \Leftrightarrow \quad f(x) \in \overline{A}\,.$$

Hence

$$x \notin A \quad \Leftrightarrow \quad f(x) \notin \overline{A}\,.$$

Hence

$$x \notin A \quad \Leftrightarrow \quad f(x) \in A\,.$$

Since A is nontrivial, and thus nonempty, we have a primitive recursive function g such that $A = \mathrm{rng}(g)$ (see Theorem 7.6.5). Thus, we have

- $x \in A \Rightarrow x \in \mathrm{rng}(g)$
- $x \notin A \Rightarrow f(x) \in \mathrm{rng}(g)$.

Let

$$h(x) \;=\; (\mu i)\,[\,g(i) = x \;\vee\; g(i) = f(x)\,]\,.$$

Observe that h is a total computable function. Let

$$\chi(x) = \begin{cases} 0 & \text{if } g(h(x)) = x \\ 1 & \text{otherwise.} \end{cases}$$

Then, χ is the characteristic function of A. Moreover, χ is a total computable function. Hence, A is a computable set.

Section 7.6.1, Exercise 11, page 244:

(b) Let A be semi-computable. There is a fixed number m such that $A = \mathcal{W}_m$. Let $f(x) = \langle x, m\rangle$. Then, $x \in A$ iff $f(x) \in \mathcal{K}_0$.

(c) This follows trivially from (b) as the set $\mathcal{K}$ is semi-computable.

(d) First we note that

$$\mathcal{K}_0 \;=\; \{\langle x,y\rangle \mid x \in \mathcal{W}_y\} \;=\; \{\langle x,y\rangle \mid \text{there exists } t \text{ such that } \mathcal{T}(x,y,t)\}\ .$$

Let $f(x,y) = 0 \;\dot{-}\; (\mu t)[\mathcal{T}((x)_1,(x)_2,t)]$. Note that $f(a,b)$ is defined if, and only if, $a \in \mathcal{K}_0$. Let $f_a(y) = f(a,y)$. The function f_a is computable (for any $a \in \mathbb{N}$). Moreover,

$$a \in \mathcal{K}_0 \quad\Rightarrow\quad \mathrm{dom}(f_a) = \mathbb{N}$$

and

$$a \notin \mathcal{K}_0 \quad\Rightarrow\quad \mathrm{dom}(f_a) = \emptyset\ .$$

If we have the number a, then we can compute an index a for the function f_a. There exists a primitive recursive function $\imath$ such that $\imath(a)$ is an index for f_a. (The function $\imath$ is given by the S-m-n Theorem.) We have

$$a \in \mathcal{K}_0 \;\Rightarrow\; \mathrm{dom}(f_a) = \mathbb{N} \;\Rightarrow\; \mathrm{dom}(\{\imath(a)\}) = \mathbb{N} \;\Rightarrow\; \imath(a) \in \mathcal{W}_{\imath(a)} \;\Rightarrow\; \imath(a) \in \mathcal{K}$$

and

$$a \notin \mathcal{K}_0 \;\Rightarrow\; \mathrm{dom}(f_a) = \emptyset \;\Rightarrow\; \mathrm{dom}(\{\imath(a)\}) = \emptyset \;\Rightarrow\; \imath(a) \notin \mathcal{W}_{\imath(a)} \;\Rightarrow\; \imath(a) \notin \mathcal{K}\ .$$

Hence, we have $a \in \mathcal{K}_0$ iff $\imath(a) \in \mathcal{K}$. By the definition of $\leq_m$, we have $\mathcal{K}_0 \leq_m \mathcal{K}$.

(e) Assume that the set A is semi-computable. By (b), we have a total computable function f_1 such that $x \in A$ iff $f_1(x) \in \mathcal{K}_0$. By (d), we have a total computable function f_2 such that $x \in K_0$ iff $f_2(x) \in \mathcal{K}$. Let $f(x) = f_2(f_1(x))$. Then, f is a total computable function such that $x \in A$ iff $f(x) \in \mathcal{K}$. Thus, A is m-reducible to $\mathcal{K}$.

Assume that A is is m-reducible to $\mathcal{K}$. Then, we have a total computable function h such that $x \in A$ iff $h(x) \in \mathcal{K}$. Let $\mathcal{K} = \mathrm{dom}(g)$. Such a computable function g exists since $\mathcal{K}$ is semi-computable. Let $f(x) = g(h(x))$. Then, f is a computable function such that $A = \mathrm{dom}(f)$. Thus, A is semi-computable.

Section 7.7.3

Section 7.7.3, Exercise 1, page 253:

(a)

(1) This is false. Let the set A of non-logical axioms be empty. Then U is a computable set. It follows from the undecidability of the Entsheidungsproblem that V is not a computable set.

(2) This is true.

(3) This is true.

(4) This is false. Let $A_0 = \{\ \overline{a} = \overline{a} \mid a \in \overline{\mathcal{K}}\ \}$. Let $A = A_0 \cup \{\ (\forall x)(\forall y)x = y\ \}$. Now, A is indeed a strange and unnatural collection of axioms. However, the set V is computable, but the set U is not (U is not even a semi-computable set).

(5) This is false. Let $A = \{\ \overline{a} = \overline{a} \mid a \in \overline{\mathcal{K}}\ \}$. Then U is not a semi-computable set. The set V is semi-computable since

$$V \ = \ \{\ \lceil \phi \rceil \mid A \vdash \phi\ \} \ = \ \{\ \lceil \phi \rceil \mid \emptyset \vdash \phi\ \}\ .$$

Section 7.7.3, Exercise 2, page 254:

Let $A = \{\ulcorner \phi \urcorner \mid \phi \text{ is a } \Sigma\text{-sentence and } \mathfrak{N} \models \phi\}$.

First we prove that A is a semi-computable set. We will use the fact that the theory N is Σ-complete, that is, for any Σ-formula θ (with no free variables), we have

$$N \vdash \theta \quad \Leftrightarrow \quad \mathfrak{N} \models \theta\ . \tag{1}$$

(The right-left implication of (1) holds by Proposition 5.3.13, and the converse implication holds by the Soundness Theorem.) Consider the Δ-formula $Deduction(y, x)$ on page 165. For any $\mathcal{L}_{NT}$-formula ϕ, we have

$$\mathfrak{N} \models (\exists y) Deduction(y, \overline{\ulcorner \phi \urcorner}) \quad \Leftrightarrow \quad N \vdash \phi\ . \tag{2}$$

Observe that $(\exists y) Deduction(y, \overline{\ulcorner \phi \urcorner})$ is a Σ-formula. By (1) and (2), we have

$$N \vdash (\exists y) Deduction(y, \overline{\ulcorner \phi \urcorner}) \quad \Leftrightarrow \quad N \vdash \phi\ . \tag{3}$$

Let $Sigma(x)$ be the predicate that holds if, and only if, x is the Gödel number of a Σ-formula. We know that $Sigma(x)$ can be defined by a Δ-formula. It is easy to see that predicates defined by Δ-formulas are primitive recursive (see Exercise 7). Hence, both $Deduction(y, x)$ and $Sigma(x)$ are primitive recursive predicates. Let

$$f(x) \ = \ (\mu y)\,[\, Deduction(y, x) \ \wedge \ Sigma(x)\,]\ .$$

Then, f is a computable function such that $A = \mathrm{dom}(f)$. Hence, A is a semi-computable set.

We will now prove that A is not a computable set. Assume for the sake of a contradiction that A is computable, and let χ_A be the characteristic function of A. By Lemma 7.7.1, we have a Σ-formula $\phi(x)$ such that $\mathfrak{N} \models \phi(\overline{a})$ iff $a \in \mathcal{K}$. Let $g(x) = \ulcorner \phi(\overline{x}) \urcorner$. The function g

is primitive recursive by Lemma 7.3.17. Let $\chi(x) = \chi_A(g(x))$. The function χ is computable. Moreover, χ is the characteristic function for the set $\mathcal{K}$. Thus, $\mathcal{K}$ is a computable set. This contradicts Theorem 7.6.12.

Section 7.7.3, Exercise 5, page 254:

Here is an informal argument: Let f be a total computable function. Assume that there does not exist an $\mathcal{L}_{NT}$-formula θ such that $f(\ulcorner\theta\urcorner) < \ell(\theta)$. Then for any $\mathcal{L}_{NT}$-formula θ, we have $\ell(\theta) \leq f(\ulcorner\theta\urcorner)$. Now we have an algorithm for deciding if a formula θ is valid:

- find the number m such that $m = f(\ulcorner\theta\urcorner) + 1$
- generate all deductions that contain fewer than m symbols
- if θ is the last formula in one of these deductions, then θ is valid; otherwise, θ is not valid.

This algorithm shows that

$$\{\ulcorner\psi\urcorner \mid \psi \text{ is a valid } \mathcal{L}_{NT}\text{-formula}\}$$

is a computable set. This contradicts Theorem 7.7.2 (Undecidability of the Entscheidungsproblem).

Here is a more formal argument: Suppose that we consider the Δ-formula $Deduction(c, f)$ on page 165. The formula $Deduction(\overline{c}, \overline{\ulcorner\phi\urcorner})$ is true (in $\mathfrak{N}$) iff c encodes a derivation of the formula ϕ. Any relation defined by Δ-formula is primitive recursive. Thus, $Deduction(y, x)$ is a primitive recursive relation (see Exercise 7).

Let $|d|$ denote the number of symbols in the derivation d. Each occurrence of a variable counts as one symbol. Without loss of generality we can assume that all of the variables occurring in the derivation d are amongst the variables in the list $v_1, v_2, \ldots, v_{|d|}$. Let g be a primitive recursive function g such that we have $c \leq g(|d|)$ if c is a number that encodes the deduction d. (Such a primitive recursive g exists if we assume that no other variables than $v_1, v_2, \ldots, v_{|d|}$ occur in d.)

Let f be a total computable function. Assume that there does not exist an $\mathcal{L}_{NT}$-formula θ such that $f(\ulcorner\theta\urcorner) < \ell(\theta)$. Then for any $\mathcal{L}_{NT}$-formula θ, we have $\ell(\theta) \leq f(\ulcorner\theta\urcorner)$. (Recall that $(\mu \leq n)[R(\vec{x}, i)]$ equals $n+1$ if there does not exist i less than or equal to n such that the relation $R(\vec{x}, i)$ holds.) Let

$$h(x) \quad = \quad (\mu i \leq g(f(x)))[\, Deduction(i, x)\,]$$

and let

$$\chi(x) = \begin{cases} 1 & \text{if } h(x) = g(f(x)) + 1 \\ 0 & \text{otherwise.} \end{cases}$$

Then, χ is a total computable function. Moreover, χ is the characteristic function of the set

$$\{\ulcorner\psi\urcorner \mid \psi \text{ is a valid } \mathcal{L}_{NT}\text{-formula}\} .$$

Thus, this set is a computable set. This contradicts Theorem 7.7.2 (Undecidability of the Entscheidungsproblem).

Section 7.8.1

Section 7.8.1, Exercise 1, page 262:
Let $p(y, x_1) = y - x_1^2$.

Section 7.8.1, Exercise 2, page 263:
Let $p(y, x_1, x_2) = y - (x_1 + 2)(x_2 + 2)$.

Section 7.8.1, Exercise 3, page 263:
Let $p(y, x_1, x_2) = y - (2x_1 + 3)x_2$.

Section 7.8.1, Exercise 5, page 263:
The set $\overline{B}$ is semi-decidable. The sets A, $\overline{A}$, B, C and $\overline{C}$ are not semi-decidable.

Section 7.8.1, Exercise 6, page 263:
(a) The set $\mathcal{K}$ is semi-decidable. Hence, by the MRDP Theorem there exists a Diophantine equation $p_0(u, y_1, \ldots, y_m) = 0$ such that

$$a \in \mathcal{K} \quad \Leftrightarrow \quad p_0(a, y_1, \ldots, y_m) = 0 \text{ has a solution in } \mathbb{N} \qquad (1)$$

Let $p(x_1, \ldots, x_n) = 0$ be an arbitrary Diophantine equation. The set

$$\{a \mid p(x_1, \ldots, x_n) = 0 \text{ has a solution in } \mathbb{N} \text{ and } a \in \mathbb{N}\}$$

is semi-computable. Thus, we have a computable index e such that

$$\mathcal{W}_e \;=\; \{a \mid p(x_1, \ldots, x_n) = 0 \text{ has a solution in } \mathbb{N} \text{ and } a \in \mathbb{N}\} .$$

Moreover,

(2a) if $p(x_1, \ldots, x_n) = 0$ has a solution in $\mathbb{N}$, then $\mathcal{W}_e = \mathbb{N}$

(2b) if $p(x_1, \ldots, x_n) = 0$ does not have a solution in $\mathbb{N}$, then $\mathcal{W}_e = \emptyset$.

Now, we have

$$
\begin{array}{ll}
p(x_1, \ldots, x_n) = 0 \text{ has a solution in } \mathbb{N} & \\
\Downarrow & \text{(2a)} \\
\mathcal{W}_e = \mathbb{N} & \\
\Downarrow & \\
e \in \mathcal{W}_e & \\
\Downarrow & \text{def. of } \mathcal{K} \\
e \in \mathcal{K} & \\
\Downarrow & \text{(1)} \\
p_0(e, y_1, \ldots, y_m) = 0 \text{ has a solution in } \mathbb{N}. &
\end{array}
$$

Furthermore,

$$
\begin{array}{ll}
p(x_1, \ldots, x_n) = 0 \text{ does not have a solution in } \mathbb{N} & \\
\Downarrow & \text{(2b)} \\
\mathcal{W}_e = \emptyset & \\
\Downarrow & \\
e \notin \mathcal{W}_e & \\
\Downarrow & \text{def. of } \mathcal{K} \\
e \notin \mathcal{K} & \\
\Downarrow & \text{(1)} \\
p_0(e, y_1, \ldots, y_m) = 0 \text{ does not have a solution in } \mathbb{N}, &
\end{array}
$$

which completes the argument.

Section 8.2, page 266

Exercise 8.2.1, page 266:

The official versions are

$$= \circ x \circ yz \circ\circ xyz$$

and

$$(\forall x)((\forall y)((\forall z)(= \circ x \circ yz \circ\circ xyz)))$$

and

$$((\neg(\neg = xy)) \ \vee \ (\neg((\neg(\neg = \circ 0x \circ 0y)) \vee (\neg(\neg = \circ 1x \circ 1y))))) \ .$$

These formulas are formulas according to our definitions. (Well, actually we should also have used $v_1, v_2, v_3, \ldots$ to denote variables.)

Exercise 8.2.2, page 267:

According to our definitions, $\mathfrak{C} \cong \mathfrak{D}$ holds if there exists a bijection $i : C \to D$ such that $i(e^{\mathfrak{C}}) = e^{\mathfrak{D}}$ and $i(0^{\mathfrak{C}}) = 0^{\mathfrak{D}}$ and $i(1^{\mathfrak{C}}) = 1^{\mathfrak{D}}$ and

$$i(c_1 \circ^{\mathfrak{C}} c_2) \ = \ i(c_1) \circ^{\mathfrak{D}} i(c_2)$$

for any $c_1, c_2 \in C$.

Let $i(n) = a^n$ for any $n \in \mathbb{N}$. Then, i is a bijection from C into D. We have $i(e^{\mathfrak{C}}) = i(0) = a^0 = \varepsilon = e^{\mathfrak{D}}$. We have $i(0^{\mathfrak{C}}) = i(1) = a^1 = a = 0^{\mathfrak{D}}$ and $i(1^{\mathfrak{C}}) = i(1) = a^1 = a = 1^{\mathfrak{D}}$. Finally, for any $c_1, c_2 \in C$, we have

$$i(c_1 \circ^{\mathfrak{C}} c_2) \ = \ i(c_1 + c_2) \ = \ a^{c_1+c_2} \ = \ a^{c_1} \circ^{\mathfrak{D}} a^{c_2} \ = \ i(c_1) \circ^{\mathfrak{D}} i(c_2) \ .$$

This proves that $\mathfrak{C} \cong \mathfrak{D}$.

Exercise 8.2.3, page 267:

We have to prove that there does not exist a bijection $i : B \to C$ such that $i(e^{\mathfrak{B}}) = e^{\mathfrak{C}}$ and $i(0^{\mathfrak{B}}) = 0^{\mathfrak{C}}$ and $i(1^{\mathfrak{B}}) = 1^{\mathfrak{C}}$ and

$$i(b_1 \circ^{\mathfrak{B}} b_2) \ = \ i(b_1) \circ^{\mathfrak{C}} i(b_2) \qquad (*)$$

for any $b_1, b_2 \in B$.

Assume for the sake of a contradiction that such an i exists. Then, we have

$$\begin{array}{lcll} i(\mathbf{01}) & = & i(\mathbf{0} \circ^{\mathfrak{B}} \mathbf{1}) & (\circ^{\mathfrak{B}} \text{ is concatenation}) \\ & = & i(\mathbf{0}) \circ^{\mathfrak{C}} i(\mathbf{1}) & (*) \\ & = & i(\mathbf{1}) \circ^{\mathfrak{C}} i(\mathbf{0}) & (\circ^{\mathfrak{C}} \text{ is addition}) \\ & = & i(\mathbf{1} \circ^{\mathfrak{B}} \mathbf{0}) & (*) \\ & = & i(\mathbf{10}) & \end{array}$$

This contradicts that i is a bijection.

Exercise 8.2.4, page 268:

- $ones(x) :\equiv \neg(\exists y)(\exists z)[x = y \circ 0 \circ z]$
- $sub(x, y) :\equiv (\exists z_1)(\exists z_2)[y = z_1 \circ x \circ z_2]$

- $\phi(x,y) :\equiv (\forall z)[(sub(z,x) \wedge ones(z)) \rightarrow sub(z,y)]$.

Exercise 8.2.5, page 269:
To get a and b elements of $\{1\}^\star$, just use *ones*. Then make sure that the length of a is less than or equal to the length of b:

$$lessthan(x,y) :\equiv ones(x) \wedge ones(y) \wedge sub(x,y).$$

Let

$$slessthan(x,y) :\equiv ones(x) \wedge ones(y) \wedge sub(x \circ 1, y)$$

or maybe

$$slessthan(x,y) :\equiv lessthan(x,y) \wedge x \neq y.$$

Exercise 8.2.6, page 269:
A relatively simple way to get this is to just say

$$even(x) :\equiv ones(x) \wedge (\exists y)(x = y \circ y).$$

There is a different way to attack this problem that might be useful as you think about subsequent exercises:

The trick for this second approach is to state that there exists a bit string of the form

$$\mathbf{0110111101111110} \ldots \mathbf{01}^i\mathbf{01}^{i+2}\mathbf{0} \ldots \mathbf{01}^{|x|}\mathbf{0}\,.$$

For example, let $even(x)$ be the formula

$$ones(x) \;\wedge\; \Big\{\, x = e \;\vee\; (\exists s)\big[\, notzz(s) \wedge init(\, 0 \circ 1 \circ 1 \circ 0\,,\, s\,) \wedge \phi(s) \wedge end(\, 0 \circ x \circ 0\,,\, s\,)\,\big] \,\Big\}$$

where $\phi(s)$ is the formula

$$(\forall t)(\forall t_1)(\forall t_2)\big[(sub(t,s) \wedge t = 0 \circ t_1 \circ 0 \circ t_2 \circ 0 \wedge ones(t_1) \wedge ones(t_2)) \rightarrow t_2 = t_1 \circ 1 \circ 1\big].$$

Exercise 8.2.7, page 269:
We need a trick similar to – but a bit more advanced than – the one we used in Exercise 8.2.6: We state that there exists a bit string of the form

$$\mathbf{001}^1\mathbf{01}^{|a|}\mathbf{001}^2\mathbf{01}^{2|a|}\mathbf{001}^3\mathbf{01}^{3|a|}\mathbf{00} \ldots \mathbf{001}^{|b|}\mathbf{01}^{|b|\cdot|a|}\mathbf{00}$$

Let

$$ones(x_1, \ldots, x_n) \;:\equiv\; ones(x_1) \wedge ones(x_2) \wedge \ldots \wedge ones(x_n)\,.$$

Let $\phi_0(x)$ be the formula

$$(\exists y_1)(\exists y_2)[x = y_1 \circ 0 \circ y_2 \ \wedge \ \mathit{ones}(y_1, y_2) \ \wedge \ y_1 \neq e \ \wedge \ y_2 \neq e] \ .$$

Then, $\phi_0(x)$ states that x is a bit string of the form $b_1\mathbf{0}b_2$ where b_1 and b_2 are nonempty strings of ones.

Let $\phi_1(x)$ be the formula

$$\mathit{init}(0 \circ 0, x) \ \wedge \ \mathit{end}(0 \circ 0, x) \ \wedge \ \neg(\exists y_1)(\exists y_2)[x = y_1 \circ 0 \circ 0 \circ 0 \circ y_2] \ .$$

Then, $\phi_1(x)$ states that x is a bit string that starts and ends with **00** and that does not contain the substring **000**.

Let $\phi(x)$ be the formula

$$\phi_1(x) \ \wedge \ (\forall u)(\forall v)\big[\big(\mathit{sub}(\, 0 \circ 0 \circ u \circ 0 \circ 0 \circ v \circ 0 \circ 0 \,,\, x\,) \\ \wedge \ \mathit{notzz}(u) \ \wedge \ \mathit{notzz}(v)\big) \ \rightarrow \ \big(\phi_0(u) \ \wedge \ \phi_0(v)\big)\big] \ .$$

Then, $\phi(x)$ states that x is of the form

$$\mathbf{00}b_1\mathbf{0}b_1'\mathbf{00}b_2\mathbf{0}b_2'\mathbf{00}b_3\mathbf{0}b_3'\ldots\mathbf{00}b_n\mathbf{0}b_n'\mathbf{00}$$

where each b, decorated or not, is a nonempty string of ones.

Let $\mathit{mult}(x, y, z)$ be the formula

$$\mathit{ones}(x, y, z) \ \wedge \ \big\{\, ((x = e \vee y = e) \wedge z = e) \ \vee \\ (\exists s)\big[\, \phi(s) \ \wedge \ \mathit{init}(\, 0 \circ 0 \circ 1 \circ 0 \circ x \circ 0 \circ 0 \,,\, s\,) \wedge \\ \mathit{end}(\, 0 \circ 0 \circ y \circ 0 \circ z \circ 0 \circ 0 \,,\, s\,) \ \wedge \ \psi(s)\big] \,\big\}$$

where $\psi(s)$ is the formula

$$(\forall v)(\forall s_1)(\forall s_2)(\forall t_1)(\forall t_2)\big[\, \big(\, \mathit{sub}(v, s) \wedge \\ v = 0 \circ 0 \circ s_1 \circ 0 \circ t_1 \circ 0 \circ 0 \circ s_2 \circ 0 \circ t_2 \circ 0 \circ 0 \wedge \\ \mathit{ones}(s_1, t_1, s_2, t_2)\big) \ \rightarrow \ \big(\, s_2 = s_1 \circ 1 \ \wedge \ t_2 = t_1 \circ x\big)\big] \ .$$

Exercise 8.2.8, page 269:

Let $\mathit{prime}(x)$ be the formula

$$\mathit{ones}(x) \ \wedge \ (\exists y)[x = y \circ 1 \circ 1] \wedge \\ (\forall y)(\forall z)[\mathit{mult}(y, z, x) \rightarrow (y = 1 \vee z = 1)] \ .$$

Exercise 8.2.9, page 269:

Let ϕ be the formula

$$(\forall x)\big[\, (\, \mathit{ones}(x) \ \wedge \ (\exists y)[x = y \circ 1 \circ 1 \circ 1] \ \wedge \ \mathit{even}(x)\,) \ \rightarrow \\ (\exists y)(\exists z)[\mathit{prime}(y) \ \wedge \ \mathit{prime}(z) \wedge \ x = y \circ z]\big] \ .$$

Exercise 8.2.10, page 270:

Let $\phi(x)$ be the formula from the solution of Exercise 8.2.7, see page 345. The formula $\phi(x)$ states that x is of the form

$$\mathbf{00}b_1\mathbf{0}b_1'\mathbf{00}b_2\mathbf{0}b_2'\mathbf{00}b_3\mathbf{0}b_3'\ldots\mathbf{00}b_n\mathbf{0}b_n'\mathbf{00}$$

where each b, decorated or not, is a nonempty string of ones.

Let $\mathit{code}(x)$ be the formula

$$\begin{aligned}\phi(x) \;\wedge\; \mathit{init}(0\circ 0\circ 1\circ 0, x) \;\wedge\;& \\ (\forall v)(\forall s_1)(\forall s_2)(\forall t_1)(\forall t_2)\big[\,\big(\,&\mathit{sub}(v,x)\wedge \\ v = 0\circ 0\circ s_1\circ 0\circ t_1\circ 0\circ 0\circ s_2\circ 0\circ t_2\circ 0\circ 0 \;\wedge\;& \\ \mathit{ones}(s_1,t_1,s_2,t_2)\big) \;\rightarrow\;& \\ s_2 = s_1\circ 1\big]\;.&\end{aligned}$$

Exercise 8.2.11, page 270:

Let $\mathit{IthElement}(x,y,z)$ be the formula

$$\mathit{ones}(x)\wedge\mathit{ones}(y)\wedge\mathit{code}(z)\wedge\mathit{sub}(0\circ 0\circ y\circ 0\circ x\circ 1\circ 0\circ 0, z).$$

Section 8.3, page 271

Exercise 8.3.1, page 271:

Let c be a constant symbol, and let $\mathcal{L}_c$ be the language $\mathcal{L}_{BT}\cup\{c\}$.

Let $\Gamma_0 = Th(\mathfrak{B})$, and let $\Gamma_{n+1} = \Gamma_n\cup\{\phi_n\}$ where

$$\phi_n \;:\equiv\; (\exists s)\;\underbrace{(0\circ(0\circ\ldots(0\circ}_{n\text{ occurrences of }0}\;e)\ldots))\circ s \;=\; c\,.$$

Let $\Gamma = \bigcup_{n\in\mathbb{N}}\Gamma_n$.

Let $\mathfrak{A}_n$ be the $\mathcal{L}_c$-structure we get when we extend $\mathfrak{B}$ by $c^{\mathfrak{A}_n} = \mathbf{0}^n$. Obviously, $\mathfrak{A}_n\models\Gamma_n$.

We will prove that Γ has a model. Let Ω be any finite subset of Γ. Since Ω is finite, we have $\Omega\subseteq\Gamma_n$ for all sufficiently large n. Thus, there exists n such that $\mathfrak{A}_n\models\Omega$, and we conclude Ω has a model. This proves that any finite subset of Γ has a model. By the Compactness Theorem Γ has a model.

Let $\mathfrak{A}$ be an $\mathcal{L}_c$-structure that is a model of Γ. Let A be the universe of $\mathfrak{A}$. In A we will find certain elements that we will call the *standard elements*. The standard elements are the interpretations of the biterals. For each $b\in\{\mathbf{0},\mathbf{1}\}^\star$, we will find an element $\bar{b}^{\mathfrak{A}}$ that is the

interpretation of the biteral $\overline{b}$. No other biteral can be interpreted as $\overline{b}^{\mathfrak{A}}$ since the sentence $\overline{b_1} \neq \overline{b_2}$ is in Γ if b_1 and b_2 are different bit strings. It is easy to see that in A we will find at least one element that is not a standard element. In A there is an element $\mathbf{c}$ such that $\mathbf{c} = c^{\mathfrak{A}}$. The sentences $\phi_0, \phi_1, \phi_2, \ldots$ are all in Γ and, hence, $\mathbf{c}$ cannot be one of the standard elements (no standard bit string starts with infinitely many zeros).

Let $\mathfrak{B}^\star$ be the $\mathcal{L}_{BT}$-structure we get when we restrict $\mathfrak{A}$ to the language $\mathcal{L}_{BT}$ (so the universe of $\mathfrak{B}^\star$ is A, and $\circ^{\mathfrak{B}^\star} = \circ^{\mathfrak{A}}$ and $0^{\mathfrak{B}^\star} = 0^{\mathfrak{A}}$ and so on).

It is easy to see that $\mathfrak{B}^*$ is elementarily equivalent to $\mathfrak{B}$: We have $\mathfrak{B}^* \models Th(\mathfrak{B})$ as $Th(\mathfrak{B}) \subseteq \Gamma$. It follows that $Th(\mathfrak{B}) = Th(\mathfrak{B}^*)$, that is, the two models are elementarily equivalent.

To show that $\mathfrak{B} \ncong \mathfrak{B}^*$, assume for the sake of a contradiction that there exists an isomorphism $i : \{\mathbf{0}, \mathbf{1}\}^\star \to A$ from $\mathfrak{B}$ into $\mathfrak{B}^*$. It is easily proven by induction on the length of $b \in \{\mathbf{0}, \mathbf{1}\}^\star$ that we have $i(\overline{b}^{\mathfrak{B}}) = \overline{b}^{\mathfrak{B}^*}$. Hence, i maps any element in the universe of $\mathfrak{B}$ to one of the standard elements in the universe of $\mathfrak{B}^*$. This contradicts that i is a bijection as the nonstandard element $\mathbf{c}$ is not in the range of i.

Exercise 8.3.2, page 271:

Let c_0 be any nonstandard element in the universe of $\mathfrak{B}^*$, and let $c_{i+1} = 1^{\mathfrak{B}^*} \circ^{\mathfrak{B}^*} c_i$.

We prove that

(1) c_i is a nonstandard element (for all $i \in \mathbb{N}$)

(2) $c_i \neq c_j$ when $i \neq j$ (for all $i, j \in \mathbb{N}$).

First we note that

$$\mathfrak{B} \models (\forall x)(\forall y)[1 \circ x = 1 \circ y \ \rightarrow \ x = y] \, .$$

Then, we also have

$$\mathfrak{B}^* \models (\forall x)(\forall y)[1 \circ x = 1 \circ y \ \rightarrow \ x = y] \qquad (*)$$

since $\mathfrak{B}$ and $\mathfrak{B}^*$ are elementarily equivalent. We have assumed that c_0 is a nonstandard element. Assume by induction hypothesis that c_m is a nonstandard element. Now, $c_{m+1} = 1^{\mathfrak{B}^*} \circ^{\mathfrak{B}^*} c_m$. It follows from (*) that c_{m+1} has to be a nonstandard element. (If c_{m+1} were a standard element, then c_m would also be a standard element.) This proves that (1) holds.

To see that (2) holds, we note that $c_m = \overline{\mathbf{1}^m}^{\mathfrak{B}^*} \circ^{\mathfrak{B}^*} c_0$ and that

$$\mathfrak{B} \models (\forall x)\overline{\mathbf{1}^m} \circ x \neq \overline{\mathbf{1}^n} \circ x$$

when $m \neq n$. Since $\mathfrak{B}$ and $\mathfrak{B}^*$ are elementarily equivalent, we have

$$\mathfrak{B}^* \models (\forall x)\overline{\mathbf{1}^m} \circ x \neq \overline{\mathbf{1}^n} \circ x$$

when $m \neq n$. Thus, for $m \neq n$, we have

$$c_m \;=\; \overline{\mathbf{1}^m}^{\mathfrak{B}^*} \circ^{\mathfrak{B}^*} c_0 \;\neq\; \overline{\mathbf{1}^n}^{\mathfrak{B}^*} \circ^{\mathfrak{B}^*} c_0 \;=\; c_n\,.$$

Section 8.4, page 271

Exercise 8.4.1, page 272:
Now, $\mathfrak{B}$ is a model for B. Thus, we have

$$B \vdash \phi \;\Rightarrow \mathfrak{B} \models \phi$$

by the Soundness Theorem.

Exercise 8.4.2, page 272:

1.	$(\forall x)(\forall y)[0 \circ x \neq 1 \circ y] \rightarrow (\forall y)[0 \circ e \neq 1 \circ y]$	(Q1)
2.	$(\forall y)[0 \circ e \neq 1 \circ y] \rightarrow [0 \circ e \neq 1 \circ e]$	(Q1)
3.	$(\forall x)(\forall y)[0 \circ x \neq 1 \circ y]$	(B5)
4.	$0 \circ e \neq 1 \circ e$	(PC) from 1,2 and 3

Exercise 8.4.3, page 272:
You will need B5 and B2.

Exercise 8.4.4, page 272:
You will need B2, B4, and B6. By B2 and B4, we have $1 \neq e$ (*). By (*) and B6, we have $1 \circ 1 \neq 1 \circ e$ (**). By (**) and B2, we get $1 \circ 1 \neq 1$.

Exercise 8.4.5, page 272:
Trick question. You do not need any nonlogical axioms at all to deduce that $1 \circ 1 = 1 \circ 1$. You can do that by using (E1) and other logical axioms.

Exercise 8.4.6, page 272:
Another trick question. The question does not make sense as it is not clear which $\mathcal{L}_{BT}$-term $1 \circ 1 \circ 1$ is meant to denote.

You will need $B3$ to deduce that, e.g., $(1 \circ (1 \circ 1)) = ((1 \circ 1) \circ 1)$. You do not need any nonlogical axioms to deduce that, e.g., $((1 \circ 1) \circ 1) = ((1 \circ 1) \circ 1)$.

Exercise 8.4.7, page 272:

We prove

$$B \vdash (y_n \circ (y_{n-1} \circ \ldots (y_1 \circ e)\ldots)) \circ t \; = \\ (y_n \circ (y_{n-1} \circ \ldots (y_1 \circ t)\ldots)) \quad (*)$$

by induction on n.

By the axiom B1, we have $B \vdash e \circ t = t$. Thus, (*) holds when $n = 0$.

Let $n > 0$. By the induction hypothesis, we have

$$B \; \vdash \; (y_{n-1} \circ \ldots (y_1 \circ e)\ldots) \circ t \quad = \quad (y_{n-1} \circ \ldots (y_1 \circ t)\ldots) \; . \quad \text{(I)}$$

By the logical axioms, we can deduce

$$B \vdash y_n \circ ((y_{n-1} \circ \ldots (y_1 \circ e)\ldots) \circ t) \; = \\ (y_n \circ (y_{n-1} \circ \ldots (y_1 \circ t)\ldots)) \quad \text{(II)}$$

from (I). By B3, we have

$$B \vdash (y_n \circ (y_{n-1} \circ \ldots (y_1 \circ e)\ldots)) \circ t \; = \\ y_n \circ ((y_{n-1} \circ \ldots (y_1 \circ e)\ldots) \circ t) \quad \text{(III)}$$

From (II) and (III), we can derive (*) by using the logical axioms.

Exercise 8.4.8, page 272:

We prove that for any variable-free $\mathcal{L}_{BT}$-term t there exists a biteral b such that $B \vdash t = b$. We will use induction on the structure of the term t.

Case $t :\equiv e$: By the logical axioms, we have $B \vdash e = e$, and e is a biteral.

Case $t :\equiv 0$: By B2, we have $B \vdash 0 = (0 \circ e)$, and $(0 \circ e)$ is a biteral.

Case $t :\equiv 1$: By B2, we have $B \vdash 1 = (1 \circ e)$, and $(1 \circ e)$ is a biteral.

Case $t :\equiv t_1 \circ t_2$: Assume by the induction hypothesis that we have biterals b_1, b_2 such that

$$B \vdash t_1 = b_1 \quad \text{(I)}$$

and

$$B \vdash t_2 = b_2 \quad \text{(II)}$$

Now, b_1 is of the form $b_1 :\equiv (c_n \circ \ldots (c_1 \circ e)\ldots)$ where $c_i \in \{0, 1\}$ for $i = 1, \ldots n$. By Exercise 8.4.7, we have

$$B \vdash (c_n \circ (c_{n-1} \circ \ldots (c_1 \circ e)\ldots)) \circ b_2 \; = \\ (c_n \circ (c_{n-1} \circ \ldots (c_1 \circ b_2)\ldots)) \; . \quad \text{(III)}$$

Let $b :\equiv (c_n \circ (c_{n-1} \circ \ldots (c_1 \circ b_2)\ldots))$. Then, b is a biteral and

$$B \vdash b_1 \circ b_2 = b \tag{IV}$$

is simply another way to write (III). By (I), (II) and the logical axioms, we have

$$B \vdash t_1 \circ t_2 = b_1 \circ b_2 \; . \tag{V}$$

By (IV), (V) and the logical axioms, we have $B \vdash t_1 \circ t_2 = b$. This completes the proof for the case $t :\equiv t_1 \circ t_2$.

Exercise 8.4.9, page 272:
Assume $\mathfrak{B} \models t_1 = t_2$ (we will prove $B \vdash t_1 = t_2$).

There exists one, and only one, biteral, b such that $\mathfrak{B} \models t_1 = b$ and $\mathfrak{B} \models t_2 = b$. By Exercise 8.4.8 and the Soundness Theorem, we have $B \vdash t_1 = b$ and $B \vdash t_2 = b$. By using logical axioms, we can deduce the formula $t_1 = t_2$ from the formulas $t_1 = b$ and $t_2 = b$. Hence, $B \vdash t_1 = t_2$.

Exercise 8.4.10, page 273:
Assume $\mathfrak{B} \models b_1 \neq b_2$.

Let n_1 denote the number of constant symbols in b_1, and let n_2 denote the number of constant symbols in b_2. We will prove $B \vdash b_1 \neq b_2$ by induction on the number $\min(n_1, n_2)$.

Assume $\min(n_1, n_2) = 1$ (Base case). Then, either b_1 or b_2 is the constant e, we have $B \vdash b_1 \neq b_2$ by B4.

Assume $\min(n_1, n_2) > 1$ (Induction step). Now we have $\ell_1, \ell_2 \in \{0, 1\}$ such that $b_1 :\equiv \ell_1 \circ a_1$ and $b_2 :\equiv \ell_2 \circ a_2$. The proof splits into two cases: (i) ℓ_1 and ℓ_2 are the same constant symbol, and (ii) ℓ_1 and ℓ_2 are different constant symbols.

Case (i): We have $\mathfrak{B} \models a_1 \neq a_2$. By our induction hypothesis, we have

$$B \vdash a_1 \neq a_2 \; . \tag{1}$$

By B6, we have

$$B \vdash a_1 \neq a_2 \;\rightarrow\; (0 \circ a_1 \neq 0 \circ a_2 \;\wedge\; 1 \circ a_1 \neq 1 \circ a_2) \; . \tag{2}$$

By (1), (2) and (PC), we have $B \vdash b_1 \neq b_2$.

Case (ii): In this case we have $B \vdash b_1 \neq b_2$ straightaway by B5. We do not need the induction hypothesis.

Exercise 8.4.11, page 273:

Assume $\mathfrak{B} \models t_1 \neq t_2$ (we will prove $B \vdash t_1 \neq t_2$).

There exists one, and only, one biteral b_1 such that $\mathfrak{B} \models t_1 = b_1$, and there exists one, and only one, biteral b_2 such that $\mathfrak{B} \models t_2 = b_2$. By Exercise 8.4.8 and the Soundness Theorem, we have $B \vdash t_1 = b_1$ and $B \vdash t_2 = b_2$.

Now, $\mathfrak{B} \models b_1 \neq b_2$ (this holds since $\mathfrak{B} \models t_1 \neq t_2$). By Exercise 8.4.10, we have $B \vdash b_1 \neq b_2$.

By using logical axioms, we can deduce the formula $t_1 \neq t_2$ from the formulas $t_1 = b_1$ and $t_2 = b_2$ and $b_1 \neq b_2$. Hence $B \vdash t_1 \neq t_2$.

Exercise 8.4.12, page 274:

Let $\phi(x_1, \ldots x_n)$ be an existential $\mathcal{L}_{BT}$-formula where all the free variables are displayed.

For any variable free $\mathcal{L}_{BT}$-terms $t_1, \ldots t_n$, we have

$$\mathfrak{B} \models \phi(t_1, \ldots t_n) \quad \Rightarrow \quad B \vdash \phi(t_1, \ldots t_n)\,. \tag{*}$$

We prove (*) by induction on the structure of ϕ. Our proof contains one case for each clause in the definition of an existential formula.

Case (i): ϕ is an atomic formula. In this case (*) holds by Exercise 8.4.9.

Case (ii): ϕ is of the form $\neg\psi$ where ψ is an atomic formula. In this case (*) holds by Exercise 8.4.11.

Case (iii): ϕ is of the form $(\alpha \wedge \beta)$. Assume $\mathfrak{B} \models (\alpha \wedge \beta)$. (We will prove $B \vdash (\alpha \wedge \beta)$.) Then, as we know that $\mathfrak{B} \models \alpha$ and $\mathfrak{B} \models \beta$, our induction hypothesis yields $B \vdash \alpha$ and $B \vdash \beta$. By (PC), we have $B \vdash (\alpha \wedge \beta)$.

Case (iv): ϕ is of the form $(\alpha \vee \beta)$. This case is similar to (iii).

Case (v): ϕ is of the form $(\exists x)(\psi)$. Now, ψ will be of the form $\psi(x, t_1, \ldots, t_m)$ where $t_1, \ldots, t_m$ are variable free $\mathcal{L}_{BT}$-terms. Assume $\mathfrak{B} \models (\exists x)\psi(x, t_1, \ldots, t_m)$. (We will prove $B \vdash (\exists x)\psi(x, t_1, \ldots, t_m)$.) There exists $b \in \{\mathbf{0}, \mathbf{1}\}^\star$ such that $\mathfrak{B} \models \psi(\overline{b}, t_1, \ldots, t_m)$. (Such a b exists since $\mathfrak{B} \models (\exists x)\psi(x, t_1, \ldots, t_m)$. Recall that $\overline{b}^{\mathfrak{B}} = b$ and that $\overline{b}$ is a variable free $\mathcal{L}_{BT}$-term.) By our induction hypothesis, we have

$$B \vdash \psi(\overline{b}, t_1, \ldots, t_m) \tag{1}$$

and

$$\psi(\overline{b}, t_1, \ldots, t_m) \;\rightarrow\; (\exists x)\psi(x, t_1, \ldots, t_m) \tag{Q2}$$

is one of the logical axioms of our proof calculus. By (1), (Q2) and (PC), we have $B \vdash (\exists x)\psi(x, t_1, \ldots, t_m)$.

It follows straightforwardly from (*) that any existential sentence true in $\mathfrak{B}$ can be deduced from the axioms of B. (There are no free variables in an existential sentence.)

Section 8.5, page 274

Exercise 8.5.1, page 274:

This follows from the Soundness Theorem.

This is one way to put it: The Soundness Theorem states that any formula deducible from B is true in all models for B. If we can find a structure $\mathfrak{A}$ such $\mathfrak{A} \models B$ and $\mathfrak{A} \not\models \phi$, we know that ϕ is not true in all models of B. By the Soundness Theorem, we can conclude that ϕ is not deducible from B.

This is another way to put it: The notation $B \models \phi$ means that ϕ is true in all models for B. The Soundness Theorem states that

$$B \vdash \phi \quad \Rightarrow \quad B \models \phi\,. \tag{*}$$

Assume $\mathfrak{A} \models B$ and $\mathfrak{A} \not\models \phi$. Then, we have $B \not\models \phi$. By $B \not\models \phi$ and (*), we have $B \not\vdash \phi$.

Exercise 8.5.2, page 274:

We will build an $\mathcal{L}_{BT}$-structure $\mathfrak{A}$ such that $\mathfrak{A} \models B$ and

$$\mathfrak{A} \not\models (\forall x)\big[\, x \neq e \;\; \rightarrow \;\; (\exists y)[0 \circ y = x \;\vee 1 \circ y = x]\,\big]\,.$$

The universe of $\mathfrak{A}$ is $\{a, \mathbf{0}, \mathbf{1}\}^\star$, that is, the set of all finite strings over the ternary alphabet $\{a, \mathbf{0}, \mathbf{1}\}$. Furthermore, $e^{\mathfrak{A}} = \varepsilon$ and $0^{\mathfrak{A}} = \mathbf{0}$ and $1^{\mathfrak{A}} = \mathbf{1}$ and $\circ^{\mathfrak{A}}$ is the standard concatenation of two strings. Now, consider any string in the universe that starts with the symbol a, e.g., the string $a\mathbf{101}$. This string is not the result of concatenating $\mathbf{0}$ with another string in the universe: We have $a\mathbf{101} \neq \mathbf{0} \circ^{\mathfrak{A}} t$ for any $t \in \{a, \mathbf{0}, \mathbf{1}\}^\star$. Neither, is the string the result of concatenating $\mathbf{1}$ with another string in the universe. Moreover, the string is not the empty string. Hence, we have

$$\mathfrak{A} \not\models\; x \neq e \;\; \rightarrow \;\; (\exists y)[0 \circ y = x \;\vee 1 \circ y = x] \;\; s[x|a\mathbf{101}]$$

for some assignment function s. Hence

$$\mathfrak{A} \not\models (\forall x)\big[\, x \neq e \;\; \rightarrow \;\; (\exists y)[0 \circ y = x \;\vee 1 \circ y = x]\,\big]\,.$$

It is easy to see that all the axioms of B are true in $\mathfrak{A}$.

Exercise 8.5.3, page 275:

It follows from the Soundness Theorem that both B_1 and B_2 are consistent. The Soundness Theorem states that

$$\Sigma \vdash \phi \;\Rightarrow\; \Sigma \models \phi\,.$$

This is equivalent to

$$\Sigma \text{ has a model} \;\Rightarrow\; \Sigma \text{ is consistent.}$$

Now, $\mathfrak{B}$ is a model for B_1. Thus, B_1 is consistent. In the previous exercise we constructed a model $\mathfrak{A}$ for B_2. Thus, B_2 is consistent.

Exercise 8.5.4, page 275:

We need infinite bit strings. Let ω denote the length of a bit string that contains an i^{th} bit for each $i \in \mathbb{N}$. (The readers familiar with ordinal numbers will know why we have picked the Greek letter ω to denote this length.) A bit string of length ω starts somewhere, that is, there is a first element, a second element, and so on, but it does not end somewhere, that is, there is no last element.

Infinite bit strings can be concatenated with finite and infinite bit strings. Let us study some examples.

Let α be the bit string of length ω that contains nothing but zeros. Then, $\mathbf{101}\alpha$ is the string where the 1^{st} bit is $\mathbf{1}$; where the 2^{nd} bit is $\mathbf{0}$; where the 3^{rd} bit is $\mathbf{1}$ and, for any $i > 3$, the i^{th} bit of $\mathbf{101}\alpha$ is $\mathbf{0}$. So if we put the finite bit string $\mathbf{101}$ in front of the infinite bit string α, we get a bit string of length ω. If we put the infinite bit string α in front of the finite bit string $\mathbf{101}$, we get a bit string that is longer than ω. For all $i > 0$, the i^{th} bit of $\alpha\mathbf{101}$ is $\mathbf{0}$. In position $\omega + 1$ of $\alpha\mathbf{101}$, we find the bit $\mathbf{1}$; in position $\omega + 2$ we find $\mathbf{0}$; and in position $\omega + 3$ we find $\mathbf{1}$. Then the string stops. The string $\alpha\mathbf{101}$ is of length $\omega + 3$.

Let α be the bit string of length ω that contains nothing but zeros, and let β be the bit string of length ω that contains nothing but ones. The string $\beta\alpha$ is a string of length $\omega + \omega$. So is the string $\alpha\alpha$. The string $\beta\alpha$ consists of an infinite sequence of ones followed by an infinite sequence of zeros, whereas the string $\alpha\alpha$ consists of an infinite sequence of zeros followed by another infinite sequence of zeros.

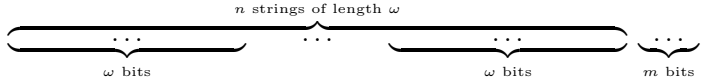
What is the length of the string $\alpha\mathbf{101}\alpha$? Well, the string is of this form

$$\underbrace{\mathbf{0000000000}\ldots}_{\omega \text{ bits}}\ \underbrace{\mathbf{1010000000}\ \ldots}_{\omega \text{ bits}}$$

so the string is of length $\omega + \omega$. The length of the string $\alpha\alpha\mathbf{10101}$ is $\omega + \omega + 5$ since the string is of the form

$$\underbrace{\mathbf{000000000}\ \ldots}_{\omega \text{ bits}}\ \underbrace{\mathbf{000000000}\ \ldots}_{\omega \text{ bits}}\ \underbrace{\mathbf{10101}}_{5 \text{ bits}}\,.$$

Let ωn denote $\overbrace{\omega + \ldots + \omega}^{n \text{ copies of } \omega}$. For any $n, m \in \mathbb{N}$, it makes sense to talk about bit strings of length $\omega n + m$. A string of length $\omega n + m$ will be of the form



Let $\mathfrak{A}$ be a $\mathcal{L}_{BT}$-structure where the universe A is

$$\{\alpha \mid \exists m, n \in \mathbb{N} \text{ and } \alpha \text{ is a bit string of length } \omega n + m\} .$$

Furthermore, let $0^{\mathfrak{A}} = \mathbf{0}$ and $1^{\mathfrak{A}} = \mathbf{1}$ and $e^{\mathfrak{A}} = \varepsilon$ (ε is the only string of length 0). Finally, let $\circ^{\mathfrak{A}}$ be concatenation (see the examples above). Then, $\mathfrak{A} \models B$. Furthermore, when α is, e.g., the bit string of length ω that contains only zeros, we have $\mathbf{0} \circ^{\mathfrak{A}} \alpha = \alpha$. Thus, $\mathfrak{A} \not\models (\forall x) 0 \circ x \neq x$.

Exercise 8.5.5, page 276:

Let $\phi(x) :\equiv 0 \circ x \neq x$. We will prove that $B_I \vdash (\forall x)\phi(x)$.

By B4, we have

$$B_I \vdash \phi(e) . \tag{1}$$

Next we need the axiom

$$(\forall x)(\forall y)[\, x \neq y \;\rightarrow\; (0 \circ x \neq 0 \circ y \;\wedge\; 1 \circ x \neq 1 \circ y)\,]. \tag{B6}$$

By this axiom, we have (use $0 \circ x$ for x, and use x for y)

$$B_I \vdash 0 \circ x \neq x \;\;\rightarrow\;\; [0 \circ (0 \circ x) \neq 0 \circ x \;\wedge\; 1 \circ (0 \circ x) \neq 1 \circ x] . \tag{2}$$

By (2) and (PC), we have

$$B_I \vdash 0 \circ x \neq x \;\;\rightarrow\;\; 0 \circ (0 \circ x) \neq 0 \circ x . \tag{3}$$

Now we need the axiom

$$(\forall x)(\forall y) 0 \circ x \neq 1 \circ y . \tag{B5}$$

By this axiom, we have (use $1 \circ x$ for x and use x for y)

$$B_I \vdash 0 \circ (1 \circ x) \neq 1 \circ x . \tag{4}$$

By (3), (4), and (PC), we have

$$B_I \vdash 0 \circ x \neq x \;\;\rightarrow\;\; [0 \circ (0 \circ x) \neq 0 \circ x \;\wedge\; 0 \circ (1 \circ x) \neq 1 \circ x] . \tag{5}$$

The formula in (5) is of the form $\phi(x) \rightarrow (\phi(0 \circ x) \;\wedge\; \phi(1 \circ x))$. Thus,

$$B_I \vdash (\forall x)[\phi(x) \;\;\rightarrow\;\; (\phi(0 \circ x) \;\wedge\; \phi(1 \circ x))] . \tag{6}$$

By (1), (6), the scheme (I), and (PC), we have $B_I \vdash (\forall x)\phi(x)$.

Section 8.6, page 276

Exercise 8.6.1, page 276:

We work through the various clauses of the inductive proof:

First, assume that f is the successor function $\mathcal{S}$. Let $\phi(x,y) :\equiv 1\circ x = y$.

Assume f is the project function $\mathcal{I}_i^n$. Let

$$\phi(x_1,\ldots,x_n,y) \;:\equiv\; x_1 = x_1 \;\wedge\; \ldots \;\wedge\; x_n = x_n \;\wedge\; x_i = y\ .$$

If f is the zero function $\mathcal{O}$, let $\phi(y) :\equiv y = 0$.

For the composition case, assume that $f(\underset{\sim}{x}) = h(g_1(\underset{\sim}{x}),\ldots,g_m(\underset{\sim}{x}))$ where $\underset{\sim}{x} = x_1,\ldots,x_n$. To improve the readability, we will assume that $n = 1$. By our induction hypothesis we have $\mathcal{L}_{BT}$-formulas $\psi_1,\ldots,\psi_m$ and ξ such that

$$g_i(a) = b \quad\Leftrightarrow\quad \mathfrak{B} \models \psi_i(\mathbf{1}^a,\bar{b}) \qquad\qquad (\text{ for } i = 1,\ldots,m)$$

and

$$h(a_1,\ldots,a_m) = b \quad\Leftrightarrow\quad \mathfrak{B} \models \xi(\overline{\mathbf{1}^{a_1}},\ldots,\overline{\mathbf{1}^{a_m}},\overline{\mathbf{1}^b})\ .$$

Let $\phi(x,y)$ be the formula

$$(\exists z_1)\ldots(\exists z_m)[\psi_1(x,z_1) \;\wedge\; \ldots \;\wedge\; \psi_i(x,z_m) \;\wedge\; \xi(z_1,\ldots,z_m,y)]\ .$$

Assume $f(\underset{\sim}{x},0) = g(\underset{\sim}{x})$ and

$$f(\underset{\sim}{x},z+1) = h(\underset{\sim}{x},z,f(\underset{\sim}{x},z))$$

where $\underset{\sim}{x} = x_1,\ldots,x_n$. To improve the readability, we will assume that $n = 1$.

The induction hypothesis yields $\mathcal{L}_{BT}$-formulas $\psi(x,y)$ and $\xi(x,z_1,z_2,y)$ such that

$$g(a) = b \quad\Leftrightarrow\quad \mathfrak{B} \models \psi(\overline{\mathbf{1}^a},\overline{\mathbf{1}^b})$$

and

$$h(a,c_1,c_2) = b \quad\Leftrightarrow\quad \mathfrak{B} \models \xi(\overline{\mathbf{1}^a},\overline{\mathbf{1}^{c_1}},\overline{\mathbf{1}^{c_2}},\overline{\mathbf{1}^b})\ .$$

Recall the formula $\mathit{slessthan}(x,y)$ from Exercise 8.2.5. Let $\phi(x,z,y)$ be the formula

$$\begin{aligned}(\exists t)(\exists u_0)\ \big[\ &\mathit{IthElement}(u_0,1,t) \;\wedge\; \psi(x,u_0) \;\wedge\; \mathit{IthElement}(y,z\circ 1,t) \;\wedge\\ &(\forall i)\ \big\{\ \mathit{slessthan}(i,z) \;\rightarrow\; (\exists u)(\exists v)[\mathit{IthElement}(u,i\circ 1,t) \;\wedge\\ &\qquad\qquad \mathit{IthElement}(v,i\circ 1\circ 1,t) \;\wedge\; \xi(x,i\circ 1,u,v)]\ \big\}\big]\ .\end{aligned}$$

Assume $f(\underset{\sim}{x}) = (\mu i)[g(\underset{\sim}{x}, i)]$ where $\underset{\sim}{x} = x_1, \ldots, x_n$. To improve the readability, we will assume that $n = 1$. By the induction hypothesis, we have an $\mathcal{L}_{BT}$-formula $\psi(x, i, y)$ such that

$$g(a, i) = b \quad \Leftrightarrow \quad \mathfrak{B} \models \psi(\overline{\mathbf{1}^a}, \overline{\mathbf{1}^i}, \overline{\mathbf{1}^b}) \ .$$

Let $\phi(x, y)$ be the formula

$$(\forall i)\big[\mathit{slessthan}(i, y) \ \rightarrow \ (\exists u)[\psi(x, i, u) \ \wedge \ \neg(u = e)]\big] \ \wedge \ \psi(x, y, e) \ .$$

(Recall that $e = \mathbf{1}^0$.)

Exercise 8.6.2, page 276:

The set $\mathcal{K}$ is semi-computable. Thus, we have a computable function f such that $\mathrm{dom}(f) = \mathcal{K}$. Let $g(x) = 0 \ \dot{-} \ f(x)$. Then, g is a computable function such that $g(a) = 0$ iff $a \in \mathcal{K}$. By Exercise 8.6.1 we have an $\mathcal{L}_{BT}$ formula $\psi(x, y)$ such that $\mathfrak{B} \models \psi(\overline{\mathbf{1}^a}, \overline{\mathbf{1}^b})$ iff $g(a) = b$. Let $\phi(x) :\equiv \psi(x, \overline{\mathbf{1}^0})$. Then, we have $\mathfrak{B} \models \phi(\overline{\mathbf{1}^a})$ iff $a \in \mathcal{K}$

Exercise 8.6.3, page 277:

Note that $\ulcorner e \urcorner = 2^9$ and that we have $\ulcorner \circ 1 t \urcorner = 2^{15}3^{13}5^{\ulcorner t \urcorner}$ for any $\mathcal{L}_{BT}$-term t. Furthermore, note that $\overline{\mathbf{1}^0} = e$ and that $\overline{\mathbf{1}^{a+1}} = \circ 1 \overline{\mathbf{1}^a}$.

Let $g(0) = 2^9$ and $g(y + 1) = 2^{15}3^{13}5^{g(y)}$. Then, we have $g(a) = \ulcorner \overline{\mathbf{1}^a} \urcorner$. The function g is defined from primitive recursive functions by composition and primitive recursion. Thus, g is a primitive recursive function.

Next we define the function f_t by induction on the structure of the $\mathcal{L}_{BT}$-term t. Let

- $f_t(a) = g(a)$ if t is the variable x
- $f_t(a) = 2^{2i}$ if t is the variable v_i and v_i is different from x
- $f_t(a) = 2^9$ if t is e
- $f_t(a) = 2^{11}$ if t is 0
- $f_t(a) = 2^{13}$ if t is 1
- $f_t(a) = 2^{15}3^{f_{t_1}(a)}5^{f_{t_2}(a)}$ if t is $\circ t_1 t_2$.

We have $f_t(a) = \ulcorner t^x_{\overline{\mathbf{1}^a}} \urcorner$ where $t^x_{\overline{\mathbf{1}^a}}$ denotes the term t where each occurrence of the variable x is replaced by the term $\overline{\mathbf{1}^a}$. For each t, the function f_t is defined by composition of primitive recursive functions. Hence, for each (fixed) term t, the function f_t is primitive recursive.

Finally, we define the function f_ϕ by induction on the structure of the formula ϕ. Let

- $f_\phi(a) = 2^1 3^{f_\alpha(a)}$ if ϕ is $(\neg\alpha)$

- $f_\phi(a) = 2^3 3^{f_\alpha(a)} 5^{f_\beta(a)}$ if ϕ is $(\alpha \vee \beta)$
- $f_\phi(a) = 2^5 3^{f_{v_i}(a)} 5^{f_\alpha(a)}$ if ϕ is $(\forall v_i)(\alpha)$
- $f_\phi(a) = 2^7 3^{f_{t_1}(a)} 5^{f_{t_2}(a)}$ if ϕ is $= t_1 t_2$.

Now, we have $f_\phi(a) = \ulcorner \phi(\overline{\mathbf{1}^a}) \urcorner$. Moreover, for each fixed formula ϕ, the function f_ϕ is defined by composition of primitive recursive functions. Thus, f_ϕ is primitive recursive.

Exercise 8.6.4, page 277:

Recall that a set is semi-computable iff it is the domain of a computable function (this is the definition of a computable set).

Since $\{\ulcorner \eta \urcorner \mid A \vdash \eta\}$ is semi-computable, we have a computable function g such that $\mathrm{dom}(g) = \{\ulcorner \eta \urcorner \mid A \vdash \eta\}$. Let f_ϕ be the primitive recursive function from Exercise 8.6.3. Then, $f_\phi(a) = \ulcorner \phi(\overline{\mathbf{1}^a}) \urcorner$ Let $h(x) = g(f_\phi(x))$. Now, h is a computable function since g and f_ϕ are computable functions. Moreover, $\mathrm{dom}(h) = \{a \mid A \vdash \phi(\overline{\mathbf{1}^a})\}$. Hence, $\{a \mid A \vdash \phi(\overline{\mathbf{1}^a})\}$ is a semi-computable set.

Exercise 8.6.5, page 278:

By Exercise 8.6.2, we have a formula $\phi(x)$ such that $\mathfrak{B} \models \phi(\overline{\mathbf{1}^a})$ iff $a \in \mathcal{K}$. Thus

$$\{a \mid \mathfrak{B} \models \neg\phi(\overline{\mathbf{1}^a})\} \tag{1}$$

is the set $\overline{\mathcal{K}}$, and we know that $\overline{\mathcal{K}}$ is not a semi-computable set. By Exercise 8.6.4, we know that

$$\{a \mid A \vdash \neg\phi(\overline{\mathbf{1}^a})\} \tag{2}$$

is a semi-computable set. This means that (1) and (2) cannot be the same set as latter is a semi-computable set whereas the former is not. We have assumed that $\mathfrak{B} \models A$. Thus, we have

$$A \vdash \neg\phi(\overline{\mathbf{1}^a}) \quad \Rightarrow \quad \mathfrak{B} \models \neg\phi(\overline{\mathbf{1}^a})$$

by the Soundness Theorem. From this we can conclude that (2) has to be a strict subset of (1). Thus, there must be a natural number a such that $\mathfrak{B} \models \neg\phi(\overline{\mathbf{1}^a})$ and $A \nvdash \neg\phi(\overline{\mathbf{1}^a})$.

Exercise 8.6.6, page 278:

It is easy to see that the set $\{\ulcorner \eta \urcorner \mid B_I \vdash \eta\}$ semi-computable. An algorithm can verify if a formula η is deducible from the axioms of B_I. The algorithm can, e.g., generate all possible B_I-deductions one by one. If the algorithm encounters a deduction of η, it halts. If the algorithm never encounters such a deduction, it runs forever. Thus, $\{\ulcorner \eta \urcorner \mid B_I \vdash \eta\}$ is a semi-computable set and, by Exercise 8.6.5, we have θ such $\mathfrak{B} \models \theta$ and $B_I \nvdash \theta$.

Exercise 8.6.7, page 278:

By the previous exercise we have θ such $\mathfrak{B} \models \theta$ and $B_I \nvdash \theta$. Moreover, we have $\mathfrak{B} \models B_I$ and $\mathfrak{B} \not\models \neg\theta$. Thus, by the Soundness Theorem, we have $B_I \nvdash \neg\theta$.

Bibliography

[Barwise 77] Jon Barwise, ed. *Handbook of Mathematical Logic.* Amsterdam: North-Holland, 1977.

[Bell and Machover 77] John L. Bell and Moshé Machover. *A Course in Mathematical Logic.* Amsterdam: North-Holland, 1977.

[Boolos 89] George Boolos. A New Proof of the Gödel Incompleteness Theorem. *Notices of the American Mathematical Society,* Vol. 36, No. 4, April 1989, pp. 388–390.

[Boolos 94] ——— Gödel's Second Incompleteness Theorem Explained in Words of One Syllable. *Mind,* Vol. 103, January 1994, pp. 1–3.

[Chang and Keisler 73] C. C. Chang and H. J. Keisler. *Model Theory.* Amsterdam: North-Holland, 1973.

[Crossley et al. 72] J. N. Crossley, C. J. Ash, C. J. Brickhill, J. C. Stillwell, and N. H. Williams. *What Is Mathematical Logic?* London: Oxford University Press, 1972.

[Dawson 97] John W. Dawson, Jr. *Logical Dilemmas: The Life and Work of Kurt Gödel.* Wellesley, Mass.: A. K. Peters, 1997.

[Enderton 72] Herbert B. Enderton. *A Mathematical Introduction to Logic.* Orlando, Fla.: Academic Press, 1972.

[Feferman 60] Solomon Feferman. Arithmetization of Metamathematics in a General Setting. *Fundamenta Mathematicae,* Vol. 49, 1960, pp. 35–92.

[Feferman 98] ——— *In the Light of Logic.* Oxford: Oxford University Press, 1998.

[Gödel–Works] Kurt Gödel. *Collected Works:* Vol. I, *Publications 1929–1936.* Edited by Soloman Feferman, John W. Dawson, Jr., Stephen C. Kleene, Gregory H. Moore, Robert M. Solovay, and Jean Van Heijenoort. New York: Oxford University Press, 1986.

[Goldstern and Judah 95] Martin Goldstern and Haim Judah. *The Incompleteness Phenomenon: A New Course in Mathematical Logic.* Wellesley, Mass.: A. K. Peters, 1995.

[Henle 86] James M. Henle. *An Outline of Set Theory.* New York: Springer-Verlag, 1986.

[Hofstadter 85] Douglas R. Hofstadter. *Metamagical Themas: Questing For The Essence Of Mind And Pattern.* Basic Books, 1985.

[Hrbacek and Jech 84] Karel Hrbacek and Thomas Jech. *Introduction to Set Theory.* New York: Marcel Dekker, 1984.

[Keisler 76] H. Jerome Keisler. *Foundations of Infinitesimal Calculus.* Boston: Prindle, Weber & Schmidt, 1976.

[Keisler and Robbin 96] H. Jerome Keisler and Joel Robbin. *Mathematical Logic and Computability.* New York: McGraw-Hill, 1996.

[Malitz 79] Jerome Malitz. *Introduction to Mathematical Logic.* New York: Springer-Verlag, 1979.

[Manin 77] Yu. I. Manin. *A Course in Mathematical Logic.* New York: Springer-Verlag, 1977.

[Mendelson 87] Elliott Mendelson. *Introduction to Mathematical Logic.* Monterey, Calif.: Brooks/Cole, a division of Wadsworth, 1987.

[Roitman 90] Judith Roitman. *Introduction to Modern Set Theory.* New York: Wiley, 1990.

[Russell 67] Bertrand Russell. *The Autobiography of Bertrand Russell,* Vol. I. London: George Allen & Unwin, 1967.

[Turing 37] A.M. Turing. On Computable Numbers, with an Application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society.* Vol. s2–42, No. 1, 1937, pp. 230–265.

Index

List of Symbols