A Mathematical Introduction to Mathematical Logic

Joseph R. Mileti

February 2, 2020

Contents

Chapter 1

Introduction

1.1 The Nature of Mathematical Logic

Mathematical logic originated as an attempt to codify and formalize the following:

1. The language of mathematics.
2. The basic assumptions of mathematics.
3. The permissible rules of proof.

One of the successful results of such a program is the ability to study mathematical language and reasoning using mathematics itself. For example, we will eventually give a precise mathematical definition of a formal proof, and to avoid confusion with our current intuitive understanding of what a proof is, we will call these objects *deductions*. One can think of our eventual definition of a deduction as analogous to the precise mathematical definition of continuity, which replaces the fuzzy "a graph that can be drawn without lifting your pencil". Once we have codified the notion in this way, we will have turned deductions into precise mathematical objects, allowing us to prove mathematical theorems about deductions using normal mathematical reasoning. For example, we will open up the possibility of proving that there is no deduction of certain mathematical statements.

Some newcomers to mathematical logic find the whole enterprise perplexing. For instance, if you come to the subject with the belief that the role of mathematical logic is to serve as a foundation to make mathematics more precise and secure, then the description above probably sounds rather circular, and this will almost certainly lead to a great deal of confusion. You may ask yourself:

> Okay, we've just given a decent definition of a deduction. However, instead of proving things about deductions following this formal definition, we're proving things about deductions using the usual informal proof style that I've grown accustomed to in other math courses. Why should I trust these informal proofs about deductions? How can we formally prove things (using deductions) about deductions? Isn't that circular? Is that why we are only giving informal proofs? I thought that I would come away from this subject feeling better about the philosophical foundations of mathematics, but we have just added a new layer to mathematics, so we now have both informal proofs and deductions, which makes the whole thing even more dubious.

Other newcomers do not see a problem. After all, mathematics is the most reliable method we have to establish truth, and there was never any serious question as to its validity. Such a person might react to the above thoughts as follows:

> We gave a mathematical definition of a deduction, so what's wrong with using mathematics to prove things about deductions? There's obviously a "real world" of true mathematics, and we are just working in that world to build a certain model of mathematical reasoning that is susceptible to mathematical analysis. It's quite cool, really, that we can subject mathematical proofs to a mathematical study by building this internal model. All of this philosophical speculation and worry about secure foundations is tiresome, and probably meaningless. Let's get on with the subject!

Should we be so dismissive of the first, philosophically inclined, student? The answer, of course, depends on your own philosophical views, but I will give my views as a mathematician specializing in logic with a definite interest in foundational questions. It is my firm belief that you should put all philosophical questions out of your mind during a first reading of the material (and perhaps forever, if you're so inclined), and come to the subject with a point of view that accepts an independent mathematical reality susceptible to the mathematical analysis you've grown accustomed to. In your mind, you should keep a careful distinction between normal "real" mathematical reasoning and the formal precise model of mathematical reasoning we are developing. Some people like to give this distinction a name by calling the normal mathematical realm we're working in the *metatheory*.

We will eventually give examples of formal theories, such as first-order set theory, which are able to support the entire enterprise of mathematics, including mathematical logic itself. Once we have developed set theory in this way, we will be able to give reasonable answers to the first student, and provide other respectable philosophical accounts of the nature of mathematics.

The ideas and techniques that were developed with philosophical goals in mind have now found application in other branches of mathematics and in computer science. The subject, like all mature areas of mathematics, has also developed its own very interesting internal questions which are often (for better or worse) divorced from its roots. Most of the subject developed after the 1930's is concerned with these internal and tangential questions, along with applications to other areas, and now foundational work is just one small, but still important, part of mathematical logic. Thus, if you have no interest in the more philosophical aspects of the subject, there remains an impressive, beautiful, and mathematically applicable theory which is worth your attention.

1.2 The Language of Mathematics

The first, and probably most essential, issue that we must address in order to provide a formal model of mathematics is how to deal with the language of mathematics. In this section, we sketch the basic ideas and motivation for the development of a language, but we will leave precise detailed definitions until later.

The first important point is that we should not use English (or any other natural language) because it is constantly changing, often ambiguous, and allows the construction of statements that are certainly not mathematical and/or arguably express very subjective sentiments. Once we've thrown out natural language, our only choice is to invent our own formal language. At first, the idea of developing a universal language seems quite daunting. How could we possibly write down one formal language that can simultaneously express the ideas in geometry, algebra, analysis, and every other field of mathematics, not to mention those we haven't developed yet? Our approach to this problem will be to avoid (consciously) doing it all at once.

Instead of starting from the bottom and trying to define primitive mathematical statements which can't be broken down further, let's first think about how to build new mathematical statements from old ones. The simplest way to do this is take already established mathematical statements and put them together using *and*, *or*, *not*, and *implies*. To keep a careful distinction between English and our language, we'll introduce symbols for each of these, and we'll call these symbols *connectives*.

1. $\wedge$ will denote *and*.

2. $\vee$ will denote *or*.

3. $\neg$ will denote *not*.

4. $\rightarrow$ will denote *implies*.

In order to ignore the nagging question of what constitutes a primitive statement, our first attempt will simply be to take an arbitrary set whose elements we think of as the primitive statements, and put them together in all possible ways using the connectives.

For example, suppose we start with the set $P = \{\mathsf{A}, \mathsf{B}, \mathsf{C}\}$. We think of A, B, and C as our primitive statements, and we may or may not care what they might express. We now want to put together the elements of P using the connectives, perhaps repeatedly. However, this naive idea quickly leads to a problem. Should the "meaning" of $\mathsf{A} \wedge \mathsf{B} \vee \mathsf{C}$ be "A holds, and either B holds or C holds", corresponding to $\mathsf{A} \wedge (\mathsf{B} \vee \mathsf{C})$, or should it be "Either both A and B holds, or C holds", corresponding to $(\mathsf{A} \wedge \mathsf{B}) \vee \mathsf{C}$? We need some way to avoid this ambiguity. Probably the most natural way to achieve this is to insert parentheses to make it clear how to group terms (we will eventually see other natural ways to overcome this issue). We now describe the collection of *formulas* of our language, denoted by $Form_P$. First, we put every element of P in $Form_P$, and then we generate other formulas using the following rules:

1. If φ and ψ are in $Form_P$, then $(\varphi \wedge \psi)$ is in $Form_P$.

2. If φ and ψ are in $Form_P$, then $(\varphi \vee \psi)$ is in $Form_P$.

3. If φ is in $Form_P$, then $(\neg\varphi)$ is in $Form_P$.

4. If φ and ψ are in $Form_P$, then $(\varphi \rightarrow \psi)$ is in $Form_P$.

Thus, the following is an element of $Form_P$:

$$((\neg(\mathsf{B} \vee ((\neg\mathsf{A}) \rightarrow \mathsf{C}))) \vee \mathsf{A}).$$

This simple setup, called *propositional logic*, is a drastic simplification of the language of mathematics, but there are already many interesting questions and theorems that arise from a careful study. We'll spend some time on it in Chapter 3.

Of course, mathematical language is much more rich and varied than what we can get using propositional logic. One important way to make more complicated and interesting mathematical statements is to make use of the quantifiers *for all* and *there exists*, which we'll denote using the symbols $\forall$ and $\exists$, respectively. In order to do so, we will need *variables* to act as something to quantify over. We'll denote variables by letters like x, y, z, etc. Once we've come this far, however, we'll have have to refine our naive notion of primitive statements above because it's unclear how to interpret a statement like $\forall\mathsf{x}\mathsf{B}$ without knowledge of the role of x "inside" B.

Let's think a little about our primitive statements. As we mentioned above, it seems daunting to come up with primitive statements for all areas of mathematics at once, so let's think of the areas in isolation. For instance, take group theory. A group is a set G equipped with a binary operation $\cdot$ (that is, $\cdot$ takes in two elements $x, y \in G$ and produces a new element of G denoted by $x \cdot y$) and an element e satisfying the following:

1. Associativity: For all $x, y, z \in G$, we have $(x \cdot y) \cdot z = x \cdot (y \cdot z)$.

2. Identity: For all $x \in G$, we have $x \cdot e = x = e \cdot x$.

3. Inverses: For all $x \in G$, there exists $y \in G$ such that $x \cdot y = e = y \cdot x$.

Although it is customary, and certainly easier on the eyes, to put $\cdot$ between two elements of the group, let's instead use the standard function notation in order to make the mathematical notation uniform across different areas. In this setting, a group is a set G equipped with a function $f\colon G \times G \to G$ and an element e satisfying the following:

1. For all $x, y, z \in G$, we have $f(f(x, y), z) = f(x, f(y, z))$.
2. For all $x \in G$, we have $f(x, e) = x = f(e, x)$.
3. For all $x \in G$, there exists $y \in G$ such that $f(x, y) = e = f(y, x)$.

In order to allow our language to make statement about groups, we introduce a *function symbol* f to represent the group operation, and a *constant symbol* e to represent the group identity. Now the group operation is supposed to take in two elements of the group, so if x and y are variables, then we should allow the formation of $\mathsf{f}(\mathsf{x}, \mathsf{y})$, which should denote an element of the group (once we've assigned elements of the group to x and y). Also, we should allow the constant symbol to be used in this way, allowing us to form things like $\mathsf{f}(\mathsf{x}, \mathsf{e})$. Once we've formed these, we should be allowed to use them like variables in more complicated expressions, such as $\mathsf{f}(\mathsf{f}(\mathsf{x}, \mathsf{e}), \mathsf{y})$. Each of these expressions formed by putting together, perhaps repeatedly, variables and the constant symbol e using the function symbol f is called a *term*. Intuitively, a term will name a certain element of the group once we've assigned elements to the variables.

With a way to name group elements in hand, we're now in position to say what out primitive statements are. The most basic thing that we can say about two group elements is whether or not they are equal, so we introduce a new *equality symbol*, which we will denote by the customary $=$. Given two terms t_1 and t_2, we call the expression $(t_1 = t_2)$ an *atomic formula*. These are our primitive statements.

With atomic formulas in hand, we can use the old connectives and the new quantifiers to make new statements. This puts us in a position to define *formulas*. First off, all atomic formulas are formulas. Given formulas we already know, we can put them together using the connectives above. Also, if φ is a formula and x is a variable then each of the following is a formula:

1. $\forall \mathsf{x} \varphi$.
2. $\exists \mathsf{x} \varphi$.

Perhaps without realizing it, we've described a reasonably powerful language capable of making many nontrivial statements. For instance, we can write formulas in this language which express the axioms for a group:

1. $\forall \mathsf{x} \forall \mathsf{y} \forall \mathsf{z} (\mathsf{f}(\mathsf{f}(\mathsf{x}, \mathsf{y}), \mathsf{z}) = \mathsf{f}(\mathsf{x}, \mathsf{f}(\mathsf{y}, \mathsf{z})))$.
2. $\forall \mathsf{x} ((\mathsf{f}(\mathsf{x}, \mathsf{e}) = \mathsf{x}) \wedge (\mathsf{f}(\mathsf{e}, \mathsf{x}) = \mathsf{x}))$.
3. $\forall \mathsf{x} \exists \mathsf{y} ((\mathsf{f}(\mathsf{x}, \mathsf{y}) = \mathsf{e}) \wedge (\mathsf{f}(\mathsf{y}, \mathsf{x}) = \mathsf{e}))$.

We can also write a formula saying that the group is abelian:

$$\forall \mathsf{x} \forall \mathsf{y} (\mathsf{f}(\mathsf{x}, \mathsf{y}) = \mathsf{f}(\mathsf{y}, \mathsf{x})),$$

along with a formula expressing that the center of the group is nontrivial:

$$\exists \mathsf{x} (\neg(\mathsf{x} = \mathsf{e}) \wedge \forall \mathsf{y} (\mathsf{f}(\mathsf{x}, \mathsf{y}) = \mathsf{f}(\mathsf{y}, \mathsf{x}))).$$

Perhaps unfortunately, we can also write syntactically correct formulas which express things nobody would ever utter, such as:

$$\forall \mathsf{x} \exists \mathsf{y} \exists \mathsf{x} (\neg(\mathsf{e} = \mathsf{e})).$$

What if you want to consider an area other than group theory? Commutative ring theory doesn't pose much of a problem, so long as we're allowed to alter the number of function symbols and constant symbols. We can simply have two function symbols a and m which take two arguments (a to represent addition and m to represent multiplication) and two constant symbols $\mathsf{0}$ and $\mathsf{1}$ ($\mathsf{0}$ to represent the additive identity and $\mathsf{1}$ to represent the multiplicative identity). Writing the axioms for commutative rings in this language is straightforward.

To take something fairly different, what about the theory of partially ordered sets? Recall that a partially ordered set is a set P equipped with a subset $\leq$ of $P \times P$, where we write $x \leq y$ to mean than (x, y) is an element of this subset, satisfying the following:

1. Reflexive: For all $x \in P$, we have $x \leq x$.
2. Antisymmetric: If $x, y \in P$ are such that $x \leq y$ and $y \leq x$, then $x = y$.
3. Transitive: If $x, y, z \in P$ are such that $x \leq y$ and $y \leq z$, then $x \leq z$.

Analogous to the syntax we used when handling the group operation, we will use notation which puts the ordering in front of the two arguments. Doing so may seem odd at this point, given that we are putting equality in the middle, but we will see that such a convention provides a unifying notation for other similar objects. We thus introduce a *relation symbol* R (intuitively representing $\leq$), and we keep the equality symbol $=$, but we no longer have a need for constant symbols or function symbols.

In this setting (without constant or function symbols), the only terms that we have (i.e. the only names for elements of the partially ordered set) are the variables. However, our atomic formulas are more interesting because now there are two basic things we can say about elements of the partial ordering: whether they are equal and whether they are related by the ordering. Thus, our atomic formulas are things of the form $t_1 = t_2$ and $\mathsf{R}(t_1, t_2)$ where t_1 and t_2 are terms. From these atomic formulas, we build up all our formulas as above.

We can now write formulas expressing the axioms of partial orderings:

1. $\forall \mathsf{x} \mathsf{R}(\mathsf{x}, \mathsf{x})$.
2. $\forall \mathsf{x} \forall \mathsf{y}((\mathsf{R}(\mathsf{x}, \mathsf{y}) \wedge \mathsf{R}(\mathsf{y}, \mathsf{x})) \rightarrow (\mathsf{x} = \mathsf{y}))$.
3. $\forall \mathsf{x} \forall \mathsf{y} \forall \mathsf{z}((\mathsf{R}(\mathsf{x}, \mathsf{y}) \wedge \mathsf{R}(\mathsf{y}, \mathsf{z})) \rightarrow \mathsf{R}(\mathsf{x}, \mathsf{z}))$.

We can also write a formula saying that the partial ordering is a linear ordering:

$$\forall \mathsf{x} \forall \mathsf{y}(\mathsf{R}(\mathsf{x}, \mathsf{y}) \vee \mathsf{R}(\mathsf{y}, \mathsf{x})),$$

along with a formula expressing that there exists a maximal element:

$$\exists \mathsf{x} \forall \mathsf{y}(\mathsf{R}(\mathsf{x}, \mathsf{y}) \rightarrow (\mathsf{x} = \mathsf{y})).$$

The general idea is that by leaving flexibility in the types and number of constant symbols, relation symbols, and function symbols, we'll be able to handle many areas of mathematics. We call this setup *first-order logic*. An analysis of first-order logic will consume the vast majority of our time.

Now we don't claim that first-order logic allows us to express everything in mathematics, nor do we claim that each of the setups above allow us to express everything of importance in that particular field. For example, take the group theory setting. We can express that every nonidentity element has order two with the formula

$$\forall \mathsf{x}(\mathsf{f}(\mathsf{x}, \mathsf{x}) = \mathsf{e}),$$

but it seems difficult to say that every element of the group has finite order. The natural guess is

$$\forall \mathsf{x} \exists \mathsf{n}(\mathsf{x}^{\mathsf{n}} = \mathsf{e}),$$

but this poses a problem for two reasons. The first is that our variables are supposed to quantify over elements of the group in question, not the natural numbers. The second is that we put no construction in our language to allow us to write something like x^{n}. For each fixed n, we can express it (for example, for $n = 3$, we can write $\mathsf{f}(\mathsf{f}(\mathsf{x},\mathsf{x}),\mathsf{x})$ and for $n = 4$, we can write $\mathsf{f}(\mathsf{f}(\mathsf{f}(\mathsf{x},\mathsf{x}),\mathsf{x}),\mathsf{x}))$, but it's not clear how to write it in a general way without allowing quantification over the natural numbers.

For another example, consider trying to express that a group is simple (i.e. has no nontrivial normal subgroups). The natural instinct is to quantify over all subsets H of the group G, and say that if it so happens that H is a normal subgroup, then H is either trivial or everything. However, we have no way to quantify over subsets. It's certainly possible to allow such constructions, and this gives *second-order logic*. We can even go further and allow quantifications over sets of subsets (for example, one way of expressing that a ring is Noetherian is to say that every nonempty set of ideals has a maximal element), which gives *third-order logic*, etc.

Newcomers to the field often find it strange that we focus primarily on first-order logic. There are many reasons to give special attention to first-order logic that we will develop throughout our study, but for now you should think of it as providing a simple example of a language which is capable of expressing many important aspects of various branches of mathematics. In fact, we'll eventually understand that the limitations of first-order logic are precisely what allow us to prove powerful theorems about it. Moreover, these powerful theorems allow us to deduce interesting mathematical consequences.

1.3 Syntax and Semantics

In the above discussion, we introduced symbols to denote certain concepts (such as using $\wedge$ in place of "and", $\forall$ in place of "for all", and a function symbol f in place of the group operation f). Building and maintaining a careful distinction between formal symbols and how to interpret them is a fundamental aspect of mathematical logic.

The basic structure of the formal statements that we write down using the symbols, connectives, and quantifiers is known as the *syntax* of the logic that we're developing. Syntax corresponds to the grammar of the language in question with no thought given to meaning. Imagine an English instructor who cared nothing for the content of your writing, but only cared that it was grammatically correct. That is exactly what the syntax of a logic is all about. Syntax is combinatorial in nature and is based on simple rules that provide admissible ways to manipulate symbols devoid of any knowledge of their intended meaning.

The manner in which we are permitted (or forced) to interpret the symbols, connectives, and quantifiers is known as the *semantics* of the the given logic. In a logic, there are often some symbols that we are forced to interpret in specific rigid way. For instance, in the above examples, we interpret the symbol $\wedge$ to mean *and*. In the propositional logic setting, this doesn't settle how to interpret a formula because we haven't said how to interpret the elements of P. We have some flexibility here, but once we assert that we should interpret certain elements of P as true and the others as false, our formulas express statements that are either true or false.

The first-order logic setting is more complicated. Since we have quantifiers, the first thing that must be done in order to interpret a formula is to fix a set X which will act as the set of objects over which the quantifiers will range. Once this is done, we can interpret each function symbol f taking k arguments as an actual function $f\colon X^k \to X$, each relation R symbol taking k arguments as a subset of X^k, and each constant symbol c as an element of X. Once we've fixed what we're talking about by provided such interpretations, we can view them as expressing something meaningful. For example, if we've fixed a group G and interpreted f as the group operation and e as the identity, then the formula

$$\forall \mathsf{x} \forall \mathsf{y} (\mathsf{f}(\mathsf{x},\mathsf{y}) = \mathsf{f}(\mathsf{y},\mathsf{x}))$$

is either true or false, according to whether G is abelian or not.

Always keep the distinction between syntax and semantics clear in your mind. Many basic theorems of the subject involve the interplay between syntax and semantics. For example, suppose that Γ is a set of formulas and that φ be a formula. We will eventually define what it means to say that Γ *implies* the formula φ. In the logics that we discuss, we will have two fundamental, but seemingly distinct, approaches. One way of saying that the formulas in Γ imply φ is semantic: whenever we provide an interpretation which makes all of the formulas of Γ true, it happens that φ is also true. For instance, if we're working in propositional logic and we have $\Gamma = \{((\mathsf{A} \wedge \mathsf{B}) \vee \mathsf{C})\}$ and $\varphi = (\mathsf{A} \vee \mathsf{C})$, then Γ implies φ in this sense because whenever we assign true/false values to A, B, and C in a way that makes the formulas in Γ true, it happens that φ will also be true. Another approach that we'll develop is syntactic. We'll define deductions which are "formal proofs" built from certain permissible syntactic manipulations, and Γ will imply φ in this sense if there is a witnessing deduction. The Soundness Theorem and the Completeness Theorem for first-order logic (and propositional logic) say that the semantic version and syntactic version are the same. This result amazingly allows one to mimic mathematical reasoning with purely syntactic manipulations.

1.4 The Point of It All

One important aspect, often mistaken as the only aspect, of mathematical logic is that it allows us to study mathematical reasoning. A prime example of this is given by the last sentence of the previous section. The Completeness Theorem says that we can capture the idea of one mathematical statement following from other mathematical statements with nothing more than syntactic rules on symbols. This is certainly computationally, philosophically, and foundationally interesting, but it's much more than that. A simple consequence of this result is the Compactness Theorem, which says something very deep about mathematical reasoning, and also has many interesting applications in mathematics.

Although we've developed the above logics with modest goals of handling certain fields of mathematics, it's a wonderful and surprising fact that we can embed (nearly) all of mathematics in an elegant and natural first-order system: first-order set theory. This opens the door to the possibility of proving that certain mathematical statements are independent of our usual axioms. In other words, there exist formulas φ such that there is no deduction (from the usual axioms) of φ, and also no deduction of $(\neg\varphi)$. Furthermore, the field of set theory has blossomed into an intricate field with its own deep and interesting questions.

Other very interesting and fundamental subjects arise when we ignore the foundational aspects and deductions altogether, and simply look at what we've accomplished by establishing a precise language to describe an area of mathematics. With a language in hand, we now have a way to say that certain objects are *definable* in that language. For instance, take the language of commutative rings mentioned above. If we fix a particular commutative ring, then the formula

$$\exists \mathsf{y}(\mathsf{m}(\mathsf{x}, \mathsf{y}) = 1)$$

has a free variable x and "defines" the set of units in the ring. With this point of view, we've opened up the possibility of proving lower bounds on the complexity of any definition of a certain object, or even of proving that no such definition exists in the given language.

Another, closely related, way to take our definitions of precise languages and run with it is the subject of *model theory*. In group theory, we state some axioms and work from there in order to study all possible realizations of the axioms, i.e. all possible groups. However, as we saw above, the group axioms arise in one possible language with one possible set of axioms. Instead, we can study all possible languages and all possible sets of axioms and see what we can prove in general and how the realizations compare to each other. In this sense, model theory is a kind of abstract abstract algebra.

Finally, although it's probably far from clear how it fits in at this point, *computability theory* is intimately related to the above subjects. To see the first glimmer of a connection, notice that computer programming languages are also formal languages with a precise grammar and a clear distinction between syntax and semantics. However, the connection runs much more deeply, as we will see in time.

1.5 Terminology, Notation, and Countable Sets

Definition 1.5.1. *We let* $\mathbb{N} = \{0, 1, 2, \dots\}$ *and we let* $\mathbb{N}^+ = \mathbb{N}\backslash\{0\}$.

Definition 1.5.2. *For each* $n \in \mathbb{N}$*, we let* $[n] = \{m \in \mathbb{N} : m < n\}$*, so* $[n] = \{0, 1, 2, \dots, n-1\}$.

We will often find a need to work with finite sequences, so we establish notation here.

Definition 1.5.3. *Let* X *be a set. Given* $n \in \mathbb{N}$*, we call a function* $\sigma\colon [n] \to X$ *a* finite sequence from X of length n*. We denote the set of all finite sequences from* X *of length* n *by* X^n*. We use* λ *to denote the unique sequence of length* 0*, so* $X^0 = \{\lambda\}$*. Finally, given a finite sequence* σ *from* X*, we use the notation* $|\sigma|$ *to mean the length of* σ.

Definition 1.5.4. *Let* X *be a set. We let* $X^* = \bigcup_{n\in\mathbb{N}} X^n$*, i.e.* X^* *is the set of all finite sequences from* X.

We denote finite sequences by simply listing the elements in order. For instance, if $X = \{a, b\}$, the sequence *aababbba* is an element of X^*. Sometimes for clarity, we'll insert commas and instead write a, a, b, a, b, b, b, a.

Definition 1.5.5. *If* $\sigma, \tau \in X^*$*, we denote the concatenation of* σ *and* τ *by* $\sigma\tau$ *or* $\sigma * \tau$.

Definition 1.5.6. *If* $\sigma, \tau \in X^*$*, we say that* σ *is an* initial segment *of* τ*, a write* $\sigma \preceq \tau$*, if* $\sigma = \tau \restriction [n]$ *for some* n*. We say that* σ *is a* proper initial segment *of* τ*, and write* $\sigma \prec \tau$ *if* $\sigma \preceq \tau$ *and* $\sigma \neq \tau$.

Definition 1.5.7. *Given a set* A*, we let* $\mathcal{P}(A)$ *be the set of all subsets of* A*, and we call* $\mathcal{P}(A)$ *the* power set *of* A.

For example, we have $\mathcal{P}(\{1, 2\}) = \{\emptyset, \{1\}, \{2\}, \{1, 2\}\}$ and $\mathcal{P}(\emptyset) = \{\emptyset\}$. A simple combinatorial argument shows that if $|A| = n$, then $|\mathcal{P}(A)| = 2^n$.

Chapter 2

Induction and Recursion

Proofs by induction and definitions by recursion are fundamental tools when working with the natural numbers. However, there are many other places where variants of these ideas apply. In fact, more delicate and exotic proofs by induction and definitions by recursion are two central tools in mathematical logic. We'll eventually see *transfinite* versions of these ideas that the provide ways to continue into strange new infinite realms, and these techniques are essential in both set theory and model theory. In this section, we develop the more modest tools of induction and recursion along structures which are generated by one-step processes, like the natural numbers. Occasionally, these types of induction and recursion are called "structural".

2.1 Induction and Recursion on $\mathbb{N}$

We begin by compiling the basic facts about induction and recursion on the natural numbers. We do not seek to prove that inductive arguments or recursive definitions on $\mathbb{N}$ are valid methods because they are "obvious" from the normal mathematical perspective which we are adopting. Besides, in order to do so, we would first have to fix a context in which we are defining $\mathbb{N}$. Eventually, we will indeed carry out such a construction in the context of axiomatic set theory, but that is not our current goal. Although the intuitive content of the results in this section are probably very familiar, our goal here is simply to carefully codify these facts in more precise ways to ease the transition to more complicated types of induction and recursion.

Definition 2.1.1. *We define* $S\colon \mathbb{N} \to \mathbb{N}$ *by letting* $S(n) = n + 1$ *for all* $n \in \mathbb{N}$.

We choose the letter S here because it is the first letter of *successor*. Induction is often stated in the following form: "If 0 has a certain property, and we know that $S(n)$ has the given property whenever n has the property, then we can conclude that every $n \in \mathbb{N}$ has the given property". We state this idea more formally using sets (and thus avoiding explicit mention of "properties") because we can always form the set $X = \{n \in \mathbb{N} : n \text{ has the given property}\}$.

Theorem 2.1.2 (Induction on $\mathbb{N}$ - Step Form)**.** *Suppose that* $X \subseteq \mathbb{N}$ *is such that* $0 \in X$ *and* $S(n) \in X$ *whenever* $n \in X$. *We then have* $X = \mathbb{N}$.

Definitions by recursion are often described informally as follows: "When defining $f(S(n))$, we are allowed to refer to the value of $f(n)$ in addition to referring to n". For instance, let $f\colon \mathbb{N} \to \mathbb{N}$ be the factorial function $f(n) = n!$. One typically sees f defined by the following recursive definition:

$$\begin{aligned} f(0) &= 1. \\ f(S(n)) &= S(n) \cdot f(n) \qquad \text{for all } n \in \mathbb{N}. \end{aligned}$$

In order to be able to generalize recursion to other situations, we aim to formalize this idea a little more abstractly and rigorously. In particular, we would prefer to avoid the direct self-reference in the *definition* of f.

Suppose then that X is a set and we're trying to define a function $f\colon \mathbb{N} \to X$ recursively. What do we need? We certainly want to know $f(0)$, and we want to have a "method" telling us how to define $f(S(n))$ from knowledge of n and the value of $f(n)$. If we want to avoid the self-referential appeal to f when invoking the value of $f(n)$, we need a method telling us what to do next regardless of the actual particular value of $f(n)$. That is, we need a method that tells us what to do on any possible value, not just the one the ends up happening to be $f(n)$. Formally, this "method" can be given by a function $g\colon \mathbb{N} \times X \to X$, which tells us what to do at the next step. Intuitively, this function acts as an iterator. That is, it says if the the last thing we were working on was input n and it so happened that we set $f(n)$ to equal $x \in A$, then we should define $f(S(n))$ to be the value $g(n, x)$.

With all this setup, we now state the theorem which says that no matter what value we want to assign to $f(0)$, and no matter what iterating function $g\colon \mathbb{N} \times X \to X$ we have, there exists a unique function $f\colon \mathbb{N} \to X$ obeying the rules.

Theorem 2.1.3 (Recursion on $\mathbb{N}$ - Step Form)**.** *Let X be a set, let $y \in X$, and let $g\colon \mathbb{N} \times X \to X$. There exists a unique function $f\colon \mathbb{N} \to X$ with the following two properties:*

1. $f(0) = y$.
2. $f(S(n)) = g(n, f(n))$ *for all* $n \in \mathbb{N}$.

In the case of the factorial function, we use $X = \mathbb{N}$, $y = 1$, and $g\colon \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ defined by $g(n, x) = S(n) \cdot x$. Theorem 2.1.3 asserts that there is a unique function $f\colon \mathbb{N} \to \mathbb{N}$ such that:

1. $f(0) = y = 1$.
2. $f(S(n)) = g(n, f(n)) = S(n) \cdot f(n)$ for all $n \in \mathbb{N}$.

Notice how we moved any mention of self-reference out of the definition of g, and pushed all of the weight onto the theorem that asserts the existence and uniqueness of a function that behaves properly, i.e. that satisfies both the initial condition and the appropriate recursive equation.

There is another version of induction on $\mathbb{N}$, sometimes called "strong induction", which appeals to the ordering of the natural numbers rather than the stepping of the successor function.

Theorem 2.1.4 (Induction on $\mathbb{N}$ - Order Form)**.** *Suppose that $X \subseteq \mathbb{N}$ is such that $n \in X$ whenever $m \in X$ for all $m \in \mathbb{N}$ with $m < n$. We then have $X = \mathbb{N}$.*

Notice that there is no need to deal with a separate base case of $n = 0$, because this situation is handled vacuously as there is no $m \in \mathbb{N}$ with $m < 0$. In other words, if we successfully prove that statement "$n \in X$ whenever $m \in X$ for all $m \in \mathbb{N}$ with $m < n$" using no additional assumptions about n, then this statement is true when $n = 0$, from which we can conclude that $0 \in X$ because the "whenever" clause is trivially true for $n = 0$. Of course, there is no harm in proving a separate base case if you are so inclined.

What about recursions that appeal to more than just the previous value? For example, consider the Fibonacci sequence $f\colon \mathbb{N} \to \mathbb{N}$ defined by $f(0) = 0$, $f(1) = 1$, and $f(n) = f(n-1) + f(n-2)$ whenever $n \geq 2$. We could certainly alter our previous version of recursion to codify the ability to look back two positions, but it is short-sighted and limiting to force ourselves to only go back a fixed finite number of positions. For example, what if we wanted to define a function f, so that if $n \geq 2$ is even, then we use $f(n/2)$ when defining $f(n)$? To handle every such possibility, we want to express the ability to use *all* smaller values. Thus, instead of having a function $g\colon \mathbb{N} \times X \to X$, where the second input codes the previous value of f, we now want to package many values together. The idea is to code all of the previous values into one finite sequence. So when defining $f(4)$, we should have access to the sequence $(f(0), f(1), f(2), f(3))$. Since we

defined sequences of length n as functions with domain $[n]$, we are really saying that we should have access to $f \restriction [4]$ when defining $f(4)$. However, to get around this self-reference, we should define our function g that will take as input an *arbitrary* finite sequence of elements of X, and tell us what to do next, assuming that this sequence is the correct code of the first n values of f. Recall that given a set X, we use X^* to denote the set of all finite sequences of elements of X.

Theorem 2.1.5 (Recursion on $\mathbb{N}$ - Order Form)**.** *Let X be a set and let $g\colon X^* \to X$. There exists a unique function $f\colon \mathbb{N} \to X$ such that*

$$f(n) = g(f \restriction [n])$$

for all $n \in \mathbb{N}$.

Notice that, in contrast to the situation in Theorem 2.1.3, we do not need to include a separate argument to g that gives the current position n. The reason why we can omit this argument is that we can always obtain n by simply taking the length of the sequence $f \restriction [n]$.

With this setup, here is how we can handle the Fibonacci numbers. Let $X = \mathbb{N}$, and defined $g\colon \mathbb{N}^* \to \mathbb{N}$ by letting

$$g(\sigma) = \begin{cases} 0 & \text{if } |\sigma| = 0 \\ 1 & \text{if } |\sigma| = 1 \\ \sigma(n-2) + \sigma(n-1) & \text{if } |\sigma| = n \text{ with } n \geq 2. \end{cases}$$

Theorem 2.1.5 asserts that there is a unique function $f\colon \mathbb{N} \to \mathbb{N}$ with $f(n) = g(f \restriction [n])$ for all $n \in \mathbb{N}$. We then have the following:

- $f(0) = g(f \restriction [n]) = g(\lambda) = 0$, where we recall that λ is the empty sequence.
- $f(1) = g(f \restriction [1]) = g(0) = 1$, where the argument 0 to g is the sequence of length 1 whose only element is 0.
- For all $n \geq 2$, we have $f(n) = g(f \restriction [n]) = f(n-2) + f(n-1)$.

2.2 Generation

There are many situations throughout mathematics where we want to look at what a certain subset "generates". For instance, we might have a subset of a group (vector space, ring, etc.), and we want to consider the subgroup (subspace, ideal, etc.) that the given subset generates. Another example is that we have a subset of the vertices of a graph, and we want to consider the set of all vertices in the graph that are reachable from the ones in the given subset. In Chapter 1, we talked about generating all formulas from primitive formulas using certain connectives. This situation will arise so frequently in what follows that it's a good idea to unify them all in a common framework.

Definition 2.2.1. *Let A be a set and let $k \in \mathbb{N}^+$. A function $h\colon A^k \to A$ is called a* k-ary function on A. *The number k is called the* arity *of the function h. A 1-ary function is sometimes called* unary *and a 2-ary function is sometimes called* binary.

Definition 2.2.2. *Suppose that A is a set, $B \subseteq A$, and $\mathcal{H}$ is a collection of functions such that each $h \in \mathcal{H}$ is a k-ary function on A for some $k \in \mathbb{N}^+$. We call $(A, B, \mathcal{H})$ a* simple generating system. *In such a situation, for each $k \in \mathbb{N}^+$, we denote the set of k-ary functions in $\mathcal{H}$ by $\mathcal{H}_k$.*

For example, let A be a group and let $B \subseteq A$ be some subset that contains the identity of A. Suppose that we want to think about the subgroup of A that is generated by B. The operations in question here are the group operation and inversion, so we let $\mathcal{H} = \{h_1, h_2\}$, whose elements are defined as follows:

1. $h_1 \colon A^2 \to A$ is given by $h_1(x, y) = x \cdot y$ for all $x, y \in A$.

2. $h_2 \colon A \to A$ is given by $h_2(x) = x^{-1}$ for all $x \in A$.

Taken together, we then have that $(A, B, \mathcal{H})$ is a simple generating system.

For another example, let V be a vector space over $\mathbb{R}$ and let $B \subseteq V$ be some subset that contains the zero vector. Suppose that we want to think about the subspace of V that is generated by B. The operations in question consist of vector addition and scalar multiplication, so we let $\mathcal{H} = \{g\} \cup \{h_r : r \in \mathbb{R}\}$ whose elements are defined as follows:

1. $g \colon V^2 \to V$ is given by $g(v, w) = v + w$ for all $v, w \in V$.

2. For each $r \in \mathbb{R}$, $h_r \colon V \to V$ is given by $h_r(v) = r \cdot v$ for all $v \in V$.

Taken together, we then have that $(V, B, \mathcal{H})$ is a simple generating system. Notice that $\mathcal{H}$ has uncountably many functions (one for each $r \in \mathbb{R}$) in this example.

There are some situations where the natural functions to put into $\mathcal{H}$ are not total, or are "multi-valued". For instance, in the first example below, we'll talk about the subfield generated by a certain subset of a field, and we'll want to include multiplicative inverses for all nonzero elements. When putting a corresponding function in $\mathcal{H}$, there is no obvious way to define it on 0.

Definition 2.2.3. *Let A be a set and let $k \in \mathbb{N}^+$. A function $h \colon A^k \to \mathcal{P}(A)$ is called a* set-valued k-ary function on A. *We call k the* arity *of h. A 1-ary set-valued function is sometimes called* unary *and a 2-ary set-valued function is sometimes called* binary.

Definition 2.2.4. *Suppose that A is a set, $B \subseteq A$, and $\mathcal{H}$ is a collection of functions such that each $h \in \mathcal{H}$ is a set-valued k-ary function on A for some $k \in \mathbb{N}^+$. We call $(A, B, \mathcal{H})$ a* generating system. *In such a situation, for each $k \in \mathbb{N}^+$, we denote the set of multi-valued k-ary functions in $\mathcal{H}$ by $\mathcal{H}_k$.*

For example, let K be a field and let $B \subseteq K$ be some subset that contains both 0 and 1. We want the subfield of K that is generated by B. The operations in question here are addition, multiplication, and both additive and multiplicative inverses. We thus let $\mathcal{H} = \{h_1, h_2, h_3, h_4\}$, whose elements are defined as follows:

1. $h_1 \colon K^2 \to \mathcal{P}(K)$ is given by $h_1(a, b) = \{a + b\}$ for all $a, b \in K$.

2. $h_2 \colon K^2 \to \mathcal{P}(K)$ is given by $h_2(a, b) = \{a \cdot b\}$ for all $a, b \in K$.

3. $h_3 \colon K \to \mathcal{P}(K)$ is given by $h_3(a) = \{-a\}$ for all $a \in K$.

4. $h_4 \colon K \to \mathcal{P}(K)$ is given by
$$h_4(a) = \begin{cases} \{a^{-1}\} & \text{if } a \neq 0 \\ \emptyset & \text{if } a = 0. \end{cases}$$

Taken together, we have that $(K, B, \mathcal{H})$ is a generating system.

For an example where we want to output multiple values, think about generating the vertices reachable from a given subset of vertices in a directed graph. Since a vertex can have many arrows coming out of it, we may want to throw in several vertices once we reach one. Suppose then that G is a directed graph with vertex set V and edge set E, and let $B \subseteq V$. We think of the edges as coded by ordered pairs, so $E \subseteq V^2$. We want to consider the subset of V reachable from B using edges from E. Thus, we want to say that if we've generated $v \in V$, and $w \in V$ is linked to v via some edge, then we should generate w. We thus let $\mathcal{H} = \{h\}$ where $h \colon V \to V$ is defined as follows:

$$h(v) = \{u \in V : (v, u) \in E\}.$$

Taken together, we have that $(V, B, \mathcal{H})$ is a generating system.

Notice that if we have a simple generating system $(A, B, \mathcal{H})$, then we can associate to it the generating system $(A, B, \mathcal{H}')$ where $\mathcal{H}' = \{h' : h \in \mathcal{H}\}$ where if $h\colon A^k \to A$ is an element of $\mathcal{H}_k$, then $h'\colon A^k \to \mathcal{P}(A)$ is defined by letting $h'(a_1, a_2, \ldots, a_k) = \{h(a_1, a_2, \ldots, a_k)\}$.

Given a generating system $(A, B, \mathcal{H})$, we want to define the set of elements of A generated from B using the functions in $\mathcal{H}$. There are many natural ways to do this. We discuss three approaches: the first approach is "from above", and the second and third are "from below". Each of these descriptions can be slightly simplified for simple generating systems, but it's not much harder to handle the more general case. Throughout, we will use the following three examples:

1. The first example is the simple generating system where $A = \mathbb{N}$, $B = \{7\}$, and $\mathcal{H} = \{h\}$ where $h\colon \mathbb{R} \to \mathbb{R}$ is the function $h(x) = 2x$.

2. The second example is the simple generating system given by the following group. Let $A = S_4$, $B = \{id, (1\ 2), (2\ 3), (3\ 4)\}$, and $\mathcal{H} = \{h_1, h_2\}$ where h_1 is the binary group operation and h_2 is the unary inverse function. Here the group operation is function composition, which happens from right to left. Thus, for example, we have $h_1((1\ 2), (2\ 3)) = (1\ 2)(2\ 3) = (1\ 2\ 3)$.

3. The third example is the generating system given by the following directed graph. Let the vertex set be $A = \{1, 2, 3, \ldots, 8\}$, and let the edge set be

$$E = \{(1,1), (1,2), (1,7), (2,8), (3,1), (4,4), (5,7), (6,1), (6,2), (6,5), (8,3)\}.$$

   In this case, let $B = \{3\}$ and $\mathcal{H} = \{h\}$ where $h\colon A \to A$ is described above for directed graphs. For example, we have $h(1) = \{1, 2, 7\}$, $h(2) = \{8\}$, and $h(7) = \emptyset$.

From Above

Our first approach is a "top-down" one. Given a generating system $(A, B, \mathcal{H})$, we want to apply the elements of $\mathcal{H}$ to tuples from B, perhaps repeatedly, until we form a kind of *closure*. Instead of thinking about the iteration, think about the final product. As mentioned, we want a set that is *closed* under the functions in $\mathcal{H}$. This idea leads to the following definition.

Definition 2.2.5. *Let $(A, B, \mathcal{H})$ be a generating system, and let $J \subseteq A$. We say that J is* inductive *if it has the following two properties:*

1. *$B \subseteq J$.*

2. *If $k \in \mathbb{N}^+$, $h \in \mathcal{H}_k$, and $a_1, a_2, \ldots, a_k \in J$, then $h(a_1, a_2, \ldots, a_k) \subseteq J$.*

Notice that if we working with a simple generating system directly (i.e. not coded as set-valued functions), then we should replace $h(a_1, a_2, \ldots, a_k) \subseteq J$ by $h(a_1, a_2, \ldots, a_k) \in J$.

Given a generating system $(A, B, \mathcal{H})$, we certainly have a trivial example of an inductive set, since we can just take A itself. Of course, we don't want just any old inductive set. Intuitively, we want the *smallest* one. Let's take a look at our three examples above in this context.

1. For the first example of a simple generating system given above (where $A = \mathbb{R}$, $B = \{7\}$, and $\mathcal{H} = \{h\}$ where $h\colon \mathbb{R} \to \mathbb{R}$ is the function $h(x) = 2x$). In this situation, each of the sets $\mathbb{R}$, $\mathbb{Z}$, $\mathbb{N}$, and $\{n \in \mathbb{N} : n$ is a multiple of $7\}$ is inductive, but none of them seem to be what we want.

2. For the group theory example, we certainly have that S_4 is an inductive set, but it's not obvious if there are any others.

3. In the directed graph example, each of the sets $\{1,2,3,5,7,8\}$, $\{1,2,3,4,7,8\}$, and $\{1,2,3,7,8\}$ is inductive, and it looks reasonable that the last one is the one we are after.

In general, if we want to talk about the *smallest* inductive subset of A, we need to prove that such an object exists. Here is where the "from above" idea comes into play. Rather than constructing a smallest inductive set directly (which might be difficult), we instead just intersect them all.

Proposition 2.2.6. *Let $(A, B, \mathcal{H})$ be a generating system. There exists a unique inductive set I such that $I \subseteq J$ for every inductive set J.*

Proof. We first prove existence. Let I be the intersection of all inductive sets, i.e.

$$I = \{a \in A : a \in J \text{ for every inductive set } J\}.$$

Directly from the definition, we know that if J is inductive, then $I \subseteq J$. Thus, we need only show that the set I is inductive.

- Let $b \in B$ be arbitrary. We have that $b \in J$ for every inductive set J (by definition of inductive), so $b \in I$. Therefore, $B \subseteq I$.

- Let $k \in \mathbb{N}^+$, $h \in \mathcal{H}_k$ and $a_1, a_2, \ldots, a_k \in I$ be arbitrary. Given any inductive set J, we have $a_1, a_2, \ldots, a_k \in J$ (since $I \subseteq J$), hence $h(a_1, a_2, \ldots, a_k) \subseteq J$ because J is inductive. Therefore, $h(a_1, a_2, \ldots, a_k) \subseteq J$ for every inductive set J, and hence $h(a_1, a_2, \ldots, a_k) \subseteq I$ by definition of I.

Putting these together, we conclude that I is inductive.

To see uniqueness, suppose that both I_1 and I_2 are inductive sets such that $I_1 \subseteq J$ and $I_2 \subseteq J$ for every inductive set J. In particular, we then must have both $I_1 \subseteq I_2$ and $I_2 \subseteq I_1$, so $I_1 = I_2$. □

Definition 2.2.7. *Let $(A, B, \mathcal{H})$ be a generating system. We denote the unique set of the previous proposition by $I(A, B, \mathcal{H})$, or simply by I when the context is clear.*

From Below: Building by Levels

The second idea is to make a system of levels, where at each new level we add the elements of A that are reachable from elements that we have already accumulated by applying an element of $\mathcal{H}$. In other words, we start with the elements of B, then apply functions from $\mathcal{H}$ to elements of B to generate (potentially) new elements. From here, we may need to apply functions from $\mathcal{H}$ again to these newly found elements to generate even more elements, etc. Notice that we still want to keep old elements around in this process, because if $h \in \mathcal{H}$ is binary, we have $b \in B$, and we generated a new a in the first round, then we will need to include $h(b, a)$ in the next round. In other words, we should keep a running tab on the elements and repeatedly apply the elements of $\mathcal{H}$ to *all* combinations generated so far in order to continue climbing up the ladder. Here is the formal definition:

Definition 2.2.8. *Let $(A, B, \mathcal{H})$ be a generating system. We define a sequence $V_n(A, B, \mathcal{H})$, or simply V_n, of subsets of A recursively as follows:*

$V_0 = B$.

$V_{n+1} = V_n \cup \{c \in A : \text{There exists } k \in \mathbb{N}^+, h \in \mathcal{H}_k, \text{ and } a_1, a_2, \ldots, a_k \in V_n \text{ such that } c \in h(a_1, a_2, \ldots, a_k)\}$.

Let $V(A, B, \mathcal{H}) = V = \bigcup_{n \in \mathbb{N}} V_n = \{a \in A : \text{There exists } n \in \mathbb{N} \text{ with } a \in V_n\}$.

Notice that if we work with a simple generating system directly (i.e. not coded as set-valued functions), then we should replace replace the definition of V_{n+1} by

$$V_{n+1} = V_n \cup \{h(a_1, a_2, \ldots, a_k) : k \in \mathbb{N}^+, h \in \mathcal{H}_k, \text{ and } a_1, a_2, \ldots, a_k \in V_n\}.$$

Let's take a look at our three examples above in this context:

1. For the first example of a simple generating system given above, we have $V_0 = B = \{7\}$. Since V_0 has only element, and the unique element h in $\mathcal{H}$ is unary with $h(7) = 14$, we have $V_1 = \{7, 14\}$. Apply h to each of these elements givens 14 and 28, so $V_2 = \{7, 14, 28\}$. From here, it is straightforward to check that $V_3 = \{7, 14, 28, 56\}$. In general, it appears that $V_n = \{7 \cdot 2^m : 0 \leq m \leq n\}$, and indeed it is possible to show this by induction on $\mathbb{N}$. From here, we can conclude that $V = \{7 \cdot 2^m : m \in \mathbb{N}\}$. See Example 2.3.2.

2. For the group theory example, we start with $V_0 = B = \{id, (1\ 2), (2\ 3), (3\ 4)\}$. To determine V_1, we add to V_0 the result of inverting all elements of V_0, and the result of multiplying pairs of elements of V_0 together. Since every element of V_0 is its own inverse, we just need to multiply *distinct* elements of V_0 together. We have $(1\ 2)(2\ 3) = (1\ 2\ 3)$, $(2\ 3)(1\ 2) = (1\ 3\ 2)$, etc. Computing all of the possibilities, we find that

   $$V_1 = \{id, (1\ 2), (2\ 3), (3\ 4), (1\ 2\ 3), (1\ 3\ 2), (1\ 2)(3\ 4), (2\ 3\ 4), (2\ 4\ 3)\}.$$

   Notice that V_1 is closed under inverses, but we now need to multiply elements of V_1 together to form new elements of V_2. For example, we have $(1\ 2)(1\ 3\ 2) = (1\ 3)$, so $(1\ 3) \in V_2$. We also have $(1\ 2)(2\ 3\ 4) = (1\ 2\ 3\ 4)$, so $(1\ 2\ 3\ 4) \in V_2$. In general, determining V_2 explicitly involves performing all of these calculations and collecting the results together. It turns out that V_3 has 20 of the 24 elements in S_4 (everything except $(1\ 4)$, $(1\ 4)(2\ 3)$, $(1\ 3\ 2\ 4)$, and $(1\ 4\ 2\ 3)$), and that $V_4 = S_4$. From here, it follows that $V_n = S_4$ for all $n \geq 4$.

3. For the directed graph example, we start with $V_0 = B = \{3\}$. Now $h(3) = \{1\}$, so $V_1 = \{1, 3\}$. Applying h to each element of V_1, we have $h(1) = \{1, 2, 7\}$ and $h(3) = \{1\}$, so $V_2 = \{1, 2, 3, 7\}$. Continuing on, we have $h(2) = \{8\}$ and $h(7) = \emptyset$, so $V_3 = \{1, 2, 3, 8\}$. At the next level, we see that $V_4 = \{1, 2, 3, 8\}$ as well, and from here it follows that $V_n = \{1, 2, 3, 8\}$ for all $n \geq 3$, and hence $V = \{1, 2, 3, 8\}$.

Proposition 2.2.9. *, Let $(A, B, \mathcal{H})$ be a generating system. If $m \leq n$, then $V_m \subseteq V_n$.*

Proof. Notice that we have $V_n \subseteq V_{n+1}$ for all $n \in \mathbb{N}$ immediately from the definition. From here, the statement follows by fixing an arbitrary m and inducting on $n \geq m$. □

Proposition 2.2.10. *Let $(A, B, \mathcal{H})$ be a generating system. For all $c \in V$, either $c \in B$ or there exists $k \in \mathbb{N}^+$, $h \in \mathcal{H}_k$, and $a_1, a_2, \ldots, a_k \in V$ with $c \in h(a_1, a_2, \ldots, a_k)$.*

Proof. Let $c \in V$ be arbitrary. Since $V = \bigcup_{n \in \mathbb{N}} V_n$, we know that there exists an $n \in \mathbb{N}$ with $c \in V_n$. By well-ordering, there is a smallest $m \in \mathbb{N}$ with $c \in V_m$. We have two cases.

- Suppose that $m = 0$. We then have $c \in V_0$, so $c \in B$.
- Suppose that $m > 0$. We then have $m - 1 \in \mathbb{N}$, and by choice of m, we know that $c \notin V_{m-1}$. By definition of V_m, this implies that there exists $k \in \mathbb{N}^+$,$h \in \mathcal{H}_k$, and $a_1, a_2, \ldots, a_k \in V_n$ such that $c \in h(a_1, a_2, \ldots, a_k)$.

□

From Below: Witnessing Sequences

The V_n construction of building a system of levels, obtained by repeatedly applying the elements of $\mathcal{H}$ to everything accumulated so far, is natural and elegant. However, the size of the levels can blow up quickly. If we want to argue that it's possible to generate a given element of A, we want be able to find a direct reason that does not involve generating all sorts of irrelevant elements along the way. Our third method therefore considers those elements of A which we are forced to put in because we see a witnessing construction.

Definition 2.2.11. *Let $(A, B, \mathcal{H})$ be a generating system. A* witnessing sequence *is an element $\sigma \in A^* \backslash \{\lambda\}$ such that for all $j < |\sigma|$, one of the following holds:*

1. *$\sigma(j) \in B$.*
2. *There exists $k \in \mathbb{N}^+$, $h \in \mathcal{H}_k$, and $i_1, i_2, \ldots, i_k < j$ such that $\sigma(j) \in h(\sigma(i_1), \sigma(i_2), \ldots, \sigma(i_k))$.*

If σ is a witnessing sequence, we call it a witnessing sequence for $\sigma(|\sigma| - 1)$ *(i.e. a witnessing sequence for the last element of that sequence).*

Notice that if we working with a simple generating system directly (i.e. not coded as set-valued functions), then we should replace replace $\sigma(j) \in h(\sigma(i_1), \sigma(i_2), \ldots, \sigma(i_k))$ with $\sigma(j) = h(\sigma(i_1), \sigma(i_2), \ldots, \sigma(i_k))$.

Definition 2.2.12. *Let $(A, B, \mathcal{H})$ be a generating system. Set*

$$W(A, B, \mathcal{H}) = W = \{a \in A : \textit{There exists a witnessing sequence for } a\}.$$

It sometimes useful to look only at those elements reachable which are witnessed by sequences of a bounded length, so for each $n \in \mathbb{N}^+$, set

$$W_n = \{a \in A : \textit{There exists a witnessing sequence for } a \textit{ of length } n\}.$$

Notice then that $W = \bigcup_{n \in \mathbb{N}^+} W_n$.

Let's take a look at our three examples above in this final context:

1. For the first example of a simple generating system given above, here's an example of a witnessing sequence is the sequence $7, 14, 28$ of length 3. Notice that the first element is in B, the second is the result of applying h to the first, and the third is the result of applying h to the second. Therefore, $28 \in W$, and in fact $28 \in W_3$. Notice that $7, 14, 7, 28$ is also a witnessing sequence for 28.

2. For the group theory example, notice that

   $$(2\ 3), (3\ 4), (2\ 3\ 4), (1\ 2), (1\ 2\ 3\ 4)$$

   is a witnessing sequence of length 5. This follows from the fact that the first, second, and fourth elements are in B, that $(2\ 3\ 4) = (2\ 3)(3\ 4)$, and that $(1\ 2\ 3\ 4) = (1\ 2)(2\ 3\ 4)$. Since we have a witnessing sequence for $(1\ 2\ 3\ 4)$, it follows that $(1\ 2\ 3\ 4) \in W$. Notice that we can extend this witnessing sequence to another as follows:

   $$(2\ 3), (3\ 4), (2\ 3\ 4), (1\ 2), (1\ 2\ 3\ 4), (1\ 3)(2\ 4).$$

   Here, we are using the fact that $(1\ 2\ 3\ 4)(1\ 2\ 3\ 4) = (1\ 3)(2\ 4)$ (notice that the i_ℓ in the definition of a witnessing sequence need not be distinct). Therefore, $(1\ 3)(2\ 4) \in W$.

3. For the directed graph example, the sequence $3, 1, 1, 1, 2, 3, 8, 3, 2$ is a witnessing sequence for 2, despite it's inefficiency.

The first simple observation is that if we truncate a witnessing sequence, what remains is a witnessing sequence.

Proposition 2.2.13. *If σ is a witnessing sequence and $|\sigma| = n$, then for all $m \in \mathbb{N}^+$ with $m < n$ we have that $\sigma \restriction [m]$ is a witnessing sequence.*

Another straightforward observation is that if we concatenate two witnessing sequences, the result is a witnessing sequence.

Proposition 2.2.14. *If σ and τ are witnessing sequences, then so is $\sigma\tau$.*

Finally, since we can always insert "dummy" elements from B (assuming it's nonempty because otherwise the result is trivial), we have the following observation.

Proposition 2.2.15. *Let $(A, B, \mathcal{H})$ be a generating system. If $m \leq n$, then $W_m \subseteq W_n$.*

Equivalence of the Definitions

We now prove that the there different constructions that we've developed all produce the same set.

Theorem 2.2.16. *Given a generating system $(A, B, \mathcal{H})$, we have*

$$I(A, B, \mathcal{H}) = V(A, B, \mathcal{H}) = W(A, B, \mathcal{H}).$$

Proof. Let $I = I(A, B, \mathcal{H})$, $V = V(A, B, \mathcal{H})$, and $W = W(A, B, \mathcal{H})$.

We first show that V is inductive, from which it follows $I \subseteq V$. Notice first that $B = V_0 \subseteq V$. Suppose now that $k \in \mathbb{N}^+$, $h \in \mathcal{H}_k$ and $a_1, a_2, \ldots, a_k \in V$. For each i with $1 \leq i \leq k$, we can fix an n_i with $a_i \in V_{n_i}$. Let $m = \max\{n_1, n_2, \ldots, n_k\}$. Using Proposition 2.2.9, we then have $a_i \in V_m$ for all i, hence $h(a_1, a_2, \ldots, a_k) \subseteq V_{m+1}$, and therefore $h(a_1, a_2, \ldots, a_k) \subseteq V$. It follows that V is inductive. By definition of I, we conclude that $I \subseteq V$.

We next show that W is inductive, from which it follows that $I \subseteq W$. Notice first that for every $b \in B$, the sequence b is a witnessing sequence, so $b \in W_1 \subseteq W$. Thus, $B \subseteq W$. Suppose now that $k \in \mathbb{N}^+$, $h \in \mathcal{H}_k$, and $a_1, a_2, \ldots, a_k \in W$. Let $c \in h(a_1, a_2, \ldots, a_k)$ be arbitrary. For each i with $1 \leq i \leq k$, we can fix a witnessing sequence σ_i for a_i. Using Proposition 2.2.14, we then have that the sequence $\sigma_1\sigma_2 \cdots \sigma_k$ obtained by concatenating all of the σ_i is a witnessing sequence. Since each of $a_1, a_2, \ldots, a_k$ appear as entries in this witnessing sequence, if we append the element c onto the end to form $\sigma_1\sigma_2 \cdots \sigma_k c$, we obtain a witnessing sequence for c. Thus, $c \in W$. Since $c \in h(a_1, a_2, \ldots, a_k)$ was arbitrary, it follows that $h(a_1, a_2, \ldots, a_k) \subseteq W$. It follows that W is inductive. By definition of I, we conclude that $I \subseteq W$.

We next show that $V_n \subseteq I$ by induction on $n \in \mathbb{N}$, from which it follows $V \subseteq I$. Notice first that $V_0 = B \subseteq I$ because I is inductive. For the inductive step, let $n \in \mathbb{N}$ be arbitrary with $V_n \subseteq I$. We show that $V_{n+1} \subseteq I$. By definition, we have

$$V_{n+1} = V_n \cup \{c \in A : \text{There exists } k \in \mathbb{N}^+, h \in \mathcal{H}_k, \text{ and } a_1, a_2, \ldots, a_k \in V_n \text{ such that } c \in h(a_1, a_2, \ldots, a_k)\}.$$

Now $V_n \subseteq I$ by the inductive hypothesis. Suppose then that $k \in \mathbb{N}^+$, $h \in \mathcal{H}_k$, and $a_1, a_2, \ldots, a_k \in V_n$, and fix an arbitrary $c \in h(a_1, a_2, \ldots, a_k)$. Since $V_n \subseteq I$, we have $a_1, a_2, \ldots, a_k \in I$, hence $h(a_1, a_2, \ldots, a_k) \subseteq I$ because I is inductive. Thus, $c \in I$. Since c was arbitrary, we conclude that $V_{n+1} \subseteq I$. By induction, $V_n \subseteq I$ for every $n \in \mathbb{N}$, hence $V \subseteq I$.

We finally show that $W_n \subseteq I$ by induction on $n \in \mathbb{N}^+$, from which it follows $W \subseteq I$. Since a witnessing sequence of length 1 must just be an element of B, we have $W_1 = B \subseteq I$ because I is inductive. For the inductive step, let $n \in \mathbb{N}^+$ be arbitrary with $W_n \subseteq I$. We show that $W_{n+1} \subseteq I$. Let σ be an arbitrary witnessing sequence of length $n+1$. By Proposition 2.2.13, we then have that that $\sigma \restriction [m+1]$ is a witnessing sequence of length $m+1$ for all $m < n$. Thus, $\sigma(m) \in W_{m+1}$ for all $m < n$. Since $W_{m+1} \subseteq W_n$ whenever

$m < n$ by Proposition 2.2.15, we conclude that $\sigma(m) \in W_n$ for all $m < n$, and hence by induction that $\sigma(m) \in I$ for all $m < n$. By definition of a witnessing sequence, we know that either $\sigma(n) \in B$ or there exists $i_1, i_2, \ldots, i_k < n$ such that $\sigma(n) = h(\sigma(i_1), \sigma(i_2), \ldots, \sigma(i_k))$. In either case, $\sigma(n) \in I$ because I is inductive. It follows that $W_{n+1} \subseteq I$. By induction, $W_n \subseteq I$ for every $n \in \mathbb{N}^+$, hence $W \subseteq I$. □

Definition 2.2.17. *Let $(A, B, \mathcal{H})$ be a generating system. We denote the common value of I, V, W by $G(A, B, \mathcal{H})$ or simply G.*

The ability the view the elements generated in three different ways is often helpful, as we can use the most convenient one when proving a theorem. For example, using Proposition 2.2.10, we obtain the following corollary.

Corollary 2.2.18. *Let $(A, B, \mathcal{H})$ be a generating system. For all $c \in G$, either $c \in B$ or there exists $k \in \mathbb{N}^+$, $h \in \mathcal{H}_k$, and $a_1, a_2, \ldots, a_k \in G$ with $c \in h(a_1, a_2, \ldots, a_k)$.*

2.3 Step Induction

Now that we have developed the idea of generation, we can formalize the concept of inductive proofs on generated sets. In this case, using our top-down definition of I makes the proof trivial.

Proposition 2.3.1 (Step Induction)**.** *Let $(A, B, \mathcal{H})$ be a generating system. Suppose that $X \subseteq A$ satisfies the following:*

1. *$B \subseteq X$.*
2. *$h(a_1, a_2, \ldots, a_k) \subseteq X$ whenever $k \in \mathbb{N}^+$, $h \in \mathcal{H}_k$, and $a_1, a_2, \ldots, a_k \in X$.*

We then have that $G \subseteq X$. Thus, if $X \subseteq G$, we have $X = G$.

Proof. Our assumption simply asserts that X is inductive, hence $G = I \subseteq X$ immediately from the definition of I. □

Notice that if we are working with a simple generating system directly (i.e. not coded as set-valued functions), then we should replace replace $h(a_1, a_2, \ldots, a_k) \subseteq X$ by $h(a_1, a_2, \ldots, a_k) \in X$. Proofs that employ Proposition 2.3.1 are simply called "proofs by induction" (on G). Proving the first statement that $B \subseteq X$ is the *base case*, where we show that all of the initial elements lie in X. Proving the second statement is the *inductive step*, where we show that if we have some elements of X and apply some h to them, the result consists entirely of elements of X.

The next example (which was our first example in each of the above constructions) illustrates how we can sometimes identify G explicitly. Notice that we use two different types of induction in the argument. One direction uses induction on $\mathbb{N}$ and the other uses induction on G as just described.

Example 2.3.2. *Consider the following simple generating system. Let $A = \mathbb{R}$, $B = \{7\}$, and $\mathcal{H} = \{h\}$ where $h \colon \mathbb{R} \to \mathbb{R}$ is the function $h(x) = 2x$. Determine G explicitly.*

Proof. As described previously, it appears that we want the set $\{7, 14, 28, 56, \ldots\}$, which we can write more formally as $\{7 \cdot 2^n : n \in \mathbb{N}\}$. Let $X = \{7 \cdot 2^n : n \in \mathbb{N}\}$.

We first show that $X \subseteq G$ by showing that $7 \cdot 2^n \in G$ for all $n \in \mathbb{N}$ by induction (on $\mathbb{N}$). We have $7 \cdot 2^0 = 7 \cdot 1 = 7 \in G$ because $B \subseteq G$, as G is inductive. Let $n \in \mathbb{N}$ be arbitrary such that $7 \cdot 2^n \in G$. Since G is inductive, we know that $h(7 \cdot 2^n) \in G$. Now $h(7 \cdot 2^n) = 2 \cdot 7 \cdot 2^n = 7 \cdot 2^{n+1}$, so $7 \cdot 2^{n+1} \in G$. Therefore, $7 \cdot 2^n \in G$ for all $n \in \mathbb{N}$ by induction on $\mathbb{N}$, and hence $X \subseteq G$.

We now show that $G \subseteq X$ by induction (on G). In other words, we use Proposition 2.3.1 by showing that X is inductive. Notice that $B \subseteq X$ because $7 = 7 \cdot 1 = 7 \cdot 2^0 \in X$. Now let $x \in X$ be arbitrary. Fix $n \in \mathbb{N}$ with $x = 7 \cdot 2^n$. We then have $h(x) = 2 \cdot x = 7 \cdot 2^{n+1} \in X$. Therefore $G \subseteq X$ by induction.

Combining both containments, we conclude that $X = G$. □

In many cases, it's very hard to give a simple explicit description of the set G. This is where induction in the form of Proposition 2.3.1 really shines, because it allows us to prove something about all elements of G despite the fact that we might have a hard time getting a handle on what exactly the elements of G look like. Here's an example.

Example 2.3.3. *Consider the following simple generating system. Let $A = \mathbb{Z}$, $B = \{6, 183\}$, and $\mathcal{H} = \{h\}$ where $h\colon A^3 \to A$ is given by $h(k, m, n) = k \cdot m + n$. Show that every element of G is divisible by 3.*

Proof. Let $X = \{n \in \mathbb{Z} : n \text{ is divisible by } 3\}$. We prove by induction that $G \subseteq X$. We first handle the base case. Notice that $6 = 3 \cdot 2$ and $183 = 3 \cdot 61$, so $B \subseteq X$.

We now do the inductive step. Suppose that $k, m, n \in X$, and fix $\ell_1, \ell_2, \ell_3 \in \mathbb{Z}$ with $k = 3\ell_1$, $m = 3\ell_2$, and $n = 3\ell_3$. We then have

$$\begin{aligned} h(k, m, n) &= k \cdot m + n \\ &= (3\ell_1) \cdot (3\ell_2) + 3\ell_3 \\ &= 9\ell_1\ell_2 + 3\ell_3 \\ &= 3(3\ell_1\ell_2 + \ell_3), \end{aligned}$$

hence $h(k, m, n) \in X$.

By induction (i.e. by Proposition 2.3.1), we have $G \subseteq X$. Thus, every element of G is divisible by 3. □

2.4 Freeness and Step Recursion

In this section, we restrict attention to simple generating systems, both for simplicity, and also because all of the examples we will use that support definition by recursion will be simple. Naively, one might expect that a straightforward analogue of Step Form of Recursion on $\mathbb{N}$ (Theorem 2.1.3) will carry over to recursion on generated sets. The hope would then be that the following is true.

Hope 2.4.1. *Suppose that $(A, B, \mathcal{H})$ is a simple generating system and X is a set. Suppose also that $\alpha\colon B \to X$ and that for every $h \in \mathcal{H}_k$, we have a function $g_h\colon (A \times X)^k \to X$. There exists a unique function $f\colon G \to X$ with the following two properties:*

1. *$f(b) = \alpha(b)$ for all $b \in B$.*
2. *$f(h(a_1, a_2, \ldots, a_k)) = g_h(a_1, f(a_1), a_2, f(a_2), \ldots, a_k, f(a_k))$ for all $h \in \mathcal{H}_k$ and all $a_1, a_2, \ldots, a_k \in G$.*

In other words, suppose that we assign initial values for the elements of B should go (via α), and we have iterating functions g_h for each $h \in \mathcal{H}$ telling us what to do with each generated element, based on what happened to the elements that generated it. Is there necessarily a unique function that satisfies the requirements? Unfortunately, this hope is too good to be true. Intuitively, we might generate an element $a \in A$ in several very different ways, and our different iterating functions conflict on what value we should assign to a. Alternatively, we might loop back and generate an element of A in multiple ways through just one function from $\mathcal{H}$. Here's a simple example to see what can go wrong.

Example 2.4.2. *Consider the following simple generating system. Let $A = \{1, 2\}$, $B = \{1\}$, and $\mathcal{H} = \{h\}$ where $h\colon A \to A$ is given by $h(1) = 2$ and $h(2) = 1$. Let $X = \mathbb{N}$. Define $\alpha\colon B \to \mathbb{N}$ by letting $\alpha(1) = 1$ and define $g_h\colon A \times \mathbb{N} \to \mathbb{N}$ by letting $g_h(a, n) = n + 1$. There is no function $f\colon G \to \mathbb{N}$ with the following two properties:*

1. *$f(b) = \alpha(b)$ for all $b \in B$.*
2. *$f(h(a)) = g_h(a, f(a))$ for all $a \in G$.*

The intuition here is that we are starting with 1, which then generates 2 via h, which then loops back around to generate 1 via h. Now $f(1)$ must agree for each of these possibilities. Here's the formal argument.

Proof. Notice first that $G = \{1, 2\}$. Suppose that $f\colon G \to \mathbb{N}$ satisfies (1) and (2) above. Since f satisfies (1), we must have $f(1) = \alpha(1) = 1$. By (2), we then have that

$$f(2) = f(h(1)) = g_h(1, f(1)) = f(1) + 1 = 1 + 1 = 2.$$

By (2) again, it follows that

$$f(1) = f(h(2)) = g_h(2, f(2)) = f(2) + 2 = 1 + 2 = 3,$$

contradicting the fact that $f(1) = 1$. □

To get around this problem, we want a definition of a "nice" simple generating system. Intuitively, we want to say something like "every element of G is generated in a unique way". The following definition is a relatively straightforward way to formulate this idea.

Definition 2.4.3. *A simple generating system $(A, B, \mathcal{H})$ is* free *if the following are true:*

1. *range$(h \restriction G^k) \cap B = \emptyset$ whenever $h \in \mathcal{H}_k$.*
2. *$h \restriction G^k$ is injective for every $h \in \mathcal{H}_k$.*
3. *range$(h_1 \restriction G^k) \cap$ range$(h_2 \restriction G^\ell) = \emptyset$ whenever $h_1 \in \mathcal{H}_k$ and $h_2 \in \mathcal{H}_\ell$ with $h_1 \neq h_2$.*

Intuitively the first property is saying that we don't loop around and generate an element of B again (like in the previous bad example), the second is saying that no element of $\mathcal{H}$ generates the same element in two different ways, and the last is saying that there do no exist two different elements of $\mathcal{H}$ that generate the same element.

Here is a straightforward example that will be useful in the next section. We will see more subtle and important examples soon.

Example 2.4.4. *Let X be a set. Consider the following simple generating system. Let $A = X^*$ be the set of all finite sequences from X, let $B = X$ (viewed as one element sequences), and let $\mathcal{H} = \{h_x : x \in X\}$ where $h_x\colon X^* \to X^*$ is the unary function $h_x(\sigma) = x\sigma$. We then have that $G = X^*\backslash\{\lambda\}$ and that $(A, B, \mathcal{H})$ is free.*

Proof. First notice that $X^*\backslash\{\lambda\}$ is inductive because $\lambda \notin B$ and $h_x(\sigma) \neq \lambda$ for all $\sigma \in X^*$. Next, a simple induction on $n \in \mathbb{N}^+$ shows that $X^n \subseteq G$ for all $n \in \mathbb{N}^+$, so $X^*\backslash\{\lambda\} \subseteq G$. It follows that $G = X^*\backslash\{\lambda\}$.

We now show that $(A, B, \mathcal{H})$ is free. We have to check the three properties:

- First notice that for any $x \in X$, we have that range$(h_x \restriction G) \cap X = \emptyset$ because every element of range$(h_x \restriction G)$ has length at least 2 (since $\lambda \notin G$).
- For any $x \in X$, we have that $h_x \restriction G$ is injective because if $h_x(\sigma) = h_x(\tau)$, then $x\sigma = x\tau$, and hence $\sigma = \tau$.
- Finally, notice that if $x, y \in X$ with $x \neq y$, we have that range$(h_x \restriction G) \cap$ range$(h_y \restriction G) = \emptyset$ because every elements of range$(h_x \restriction G)$ begins with x while every element of range$(h_y \restriction G)$ begins with y.

Therefore, $(A, B, \mathcal{H})$ is free. □

On to the theorem saying that if a simple generating system is free, then we can perform recursive definitions on the elements that are generated.

Theorem 2.4.5. *Suppose that the simple generating system $(A, B, \mathcal{H})$ is free and X is a set. Let $\alpha\colon B \to X$, and for each $h \in \mathcal{H}_k$, let $g_h\colon (A \times X)^k \to X$. There exists a unique function $f\colon G \to X$ with the following two properties:*

1. *$f(b) = \alpha(b)$ for all $b \in B$.*
2. *$f(h(a_1, a_2, \ldots, a_k)) = g_h(a_1, f(a_1), a_2, f(a_2), \ldots, a_k, f(a_k))$ for all $h \in \mathcal{H}_k$ and all $a_1, a_2, \ldots, a_k \in G$.*

It turns out that the uniqueness part of the theorem follows by a reasonably straightforward induction on G, and in fact does not require the assumption that $(A, B, \mathcal{H})$ is free. The hard part is proving existence. We need to define an f, and so we need to take an arbitrary a in A and determine where to send it. How can we do that? The basic idea is to build a new simple generating system whose elements are pairs (a, x) where $a \in A$ and $x \in X$. Intuitively, we want to generate the pair (a, x) if something (either α or one of the g_h functions) tells us that we need to set $f(a) = x$ if we want to satisfy the above conditions. We then go on to prove (by induction on G) that for every $a \in A$, there exists a unique $x \in X$ such that (a, x) is in our new generating system. Thus, there are no conflicts, so we can use this to define our function. In other words, we watch the generation of elements of G happen, and carry along added information telling us where we need to send the elements as we generate them. Now for the details.

Proof. Let $A' = A \times X$, $B' = \{(b, \alpha(b)) : b \in B\} \subseteq A'$, and $\mathcal{H}' = \{g'_h : h \in \mathcal{H}\}$ where for each $h \in \mathcal{H}_k$, the function $g'_h\colon (A \times X)^k \to A \times X$ is given by

$$g'_h(a_1, x_1, a_2, x_2, \ldots, a_k, x_k) = (h(a_1, a_2, \ldots, a_k), g_h(a_1, x_1, a_2, x_2, \ldots, a_k, x_k)).$$

Let $G' = G(A', B', \mathcal{H}')$. A straightforward induction (on G'), which we omit, shows that if $(a, x) \in G'$, then $a \in G$. Let

$$Z = \{a \in G : \text{There exists a unique } x \in X \text{ such that } (a, x) \in G'\}.$$

We prove by induction (on G) that $Z = G$.

- *Base Case:* Notice that for each $b \in B$, we have $(b, \alpha(b)) \in B' \subseteq G'$, hence there exists an $x \in X$ such that $(b, x) \in G'$. Let $b \in B$ be arbitrary, and suppose that $y \in X$ is such that $(b, y) \in G'$ and $y \neq \alpha(b)$. We then have $(b, y) \notin B'$, hence by Corollary 2.2.18 there exists $h \in \mathcal{H}_k$ and $(a_1, x_1), (a_2, x_2), \ldots, (a_k, x_k) \in G'$ such that

  $$\begin{aligned}(b, y) &= g'_h(a_1, x_1, a_2, x_2, \ldots, a_k, x_k)\\ &= (h(a_1, a_2, \ldots, a_k), g_h(a_1, x_1, a_2, x_2, \ldots, a_k, x_k)).\end{aligned}$$

  In particular, we then have $b = h(a_1, a_2, \ldots, a_k)$. Since $a_1, a_2, \ldots, a_k \in G$, this contradicts the fact that $\text{range}(h \restriction G^k) \cap B = \emptyset$. Therefore, for every $b \in B$, there exists a unique $x \in X$, namely $\alpha(b)$, such that $(b, x) \in G'$. Hence, $B \subseteq Z$.

- *Inductive Step:* Let $h \in \mathcal{H}_k$ and $a_1, a_2, \ldots, a_k \in Z$ be arbitrary. For each i, let x_i be the unique element of X with $(a_i, x_i) \in G'$. Since G' is inductive, we have that

  $$g'_h(a_1, x_1, a_2, x_2, \ldots, a_k, x_k) \in G',$$

  which means that

  $$(h(a_1, a_2, \ldots, a_k), g_h(a_1, x_1, a_2, x_2, \ldots, a_k, x_k)) = g'_h(a_1, x_1, a_2, x_2, \ldots, a_k, x_k) \in G'.$$

  Thus, there exists $x \in X$ such that $(h(a_1, a_2, \ldots, a_k), x) \in G'$. Suppose now that $y \in X$ is such that $(h(a_1, a_2, \ldots, a_k), y) \in G'$. We have $(h(a_1, a_2, \ldots, a_k), y) \notin B'$ because $\text{range}(h \restriction G^k) \cap B = \emptyset$, hence by Corollary 2.2.18 there exists $\hat{h} \in \mathcal{H}_\ell$ together with $(c_1, z_1), (c_2, z_2), \ldots, (c_\ell, z_\ell) \in G'$ such that

  $$\begin{aligned}(h(a_1, a_2, \ldots, a_k), y) &= g'_{\hat{h}}(c_1, z_1, c_2, z_2, \ldots, c_k, z_k)\\ &= (\hat{h}(c_1, c_2, \ldots, c_\ell), g_{\hat{h}}(c_1, z_1, c_2, z_2, \ldots, c_\ell, z_\ell)).\end{aligned}$$

In particular, we have $h(a_1, a_2, \ldots, a_k) = \hat{h}(c_1, c_2, \ldots, c_\ell)$. Since $c_1, c_2, \ldots, c_\ell \in G$, it follows that $h = \hat{h}$ because $\text{range}(h \restriction G^k) \cap \text{range}(\hat{h} \restriction G^\ell) = \emptyset$ if $h \neq \hat{h}$. Since $h = \hat{h}$, we also have $k = \ell$. Now using the fact that $h \restriction G^k$ is injective, we conclude that $a_i = c_i$ for all i. Therefore,

$$y = g_{\hat{h}}(c_1, z_1, c_2, z_2, \ldots, c_\ell, z_\ell) = g_h(a_1, x_1, a_2, x_2, \ldots, a_k, x_k).$$

Hence, there exists a unique $x \in X$, namely $g_h(a_1, x_1, a_2, x_2, \ldots, a_k, x_k)$, such that $(h(a_1, a_2, \ldots, a_k), x) \in G'$.

It now follows by induction that $Z = G$.

Define $f\colon G \to X$ by letting $f(a)$ be the unique $x \in X$ such that $(a, x) \in G'$. We need to check that f satisfies the required conditions. As stated above, for each $b \in B$, we have $(b, \alpha(b)) \in G'$, so $f(b) = \alpha(b)$. Thus, f satisfies condition (1). Now let $h \in \mathcal{H}_k$ and $a_1, a_2, \ldots, a_k \in G$ be arbitrary. We have $(a_i, f(a_i)) \in G'$ for all i, hence

$$(h(a_1, a_2, \ldots, a_k), g_h(a_1, f(a_1), a_2, f(a_2), \ldots, a_k, f(a_k))) \in G'$$

by the argument in the inductive step above (since $G = Z$). It follows that

$$f(h(a_1, a_2, \ldots, a_k)) = g_h(a_1, f(a_1), a_2, f(a_2), \ldots, a_k, f(a_k)),$$

so f also satisfies condition (2).

Finally, we need to show that f is unique. Suppose that $f_1, f_2\colon G \to X$ satisfy the conditions (1) and (2). Let $Y = \{a \in G : f_1(a) = f_2(a)\}$. We show that $Y = G$ by induction on G. First notice that for any $b \in B$ we have

$$f_1(b) = \alpha(b) = f_2(b),$$

hence $b \in Y$. It follows that $B \subseteq Y$. Now let $h \in \mathcal{H}_k$ and $a_1, a_2, \ldots, a_k \in Y$ be arbitrary. Since $a_i \in Y$ for each i, we have $f_1(a_i) = f_2(a_i)$ for each i, and hence

$$\begin{aligned} f_1(h(a_1, a_2, \ldots, a_k)) &= g_h(a_1, f_1(a_1), a_2, f_1(a_2), \ldots, a_k, f_1(a_k)) \\ &= g_h(a_1, f_2(a_1), a_2, f_2(a_2), \ldots, a_k, f_2(a_k)) \\ &= f_2(h(a_1, a_2, \ldots, a_k)). \end{aligned}$$

Thus, $h(a_1, a_2, \ldots, a_k) \in Y$. It follows by induction that $Y = G$, hence $f_1(a) = f_2(a)$ for all $a \in G$. □

2.5 An Illustrative Example

We now embark on a careful formulation and proof of the following statement: If $f\colon A^2 \to A$ is associative, i.e. $f(a, f(b, c)) = f(f(a, b), c)$ for all $a, b, c \in A$, then any "grouping" of terms which preserves the ordering of the elements inside the grouping gives the same value. In particular, if we are working in a group A, then we can write things like *acabba* without parentheses, because any allowable insertion of parentheses gives the same value. Of course, this result is not terribly surprising, and you've likely made extensive use of it when doing algebra. However, a careful proof is a bit tricky, simply because it's not immediately obvious how to define "an allowable insertion of parentheses" in a rigorous way.

Throughout this section, let A be a set not containing the symbols [,], or $\star$. The symbols [and] will code our parentheses, and $\star$ will code the application of our function. We choose to use square brackets for our parentheses to distinguish them from the normal parentheses we will use as in our mathematical reasoning. For example, we will plug these symbols into normal mathematical functions, and writing something like $g(()$ is much more confusing than $g([)$. In other words, we want to distinguish between the parentheses in our expressions and the parentheses we use in the mathematical metatheory to study the expressions. We will also use infix notation with $\star$ as usual, rather than the standard function notation, i.e. we will write $[a \star b]$ in place of $f(a, b)$.

Let $Sym_A = A \cup \{[,], \star\}$, so Sym_A is the collection of symbols that we are allowed to work with. When thinking about the expressions that we can form using the symbols in Sym_A, we want to treat everything in a purely syntactic manner. That is, we just want to write down valid sequence of symbols, without thinking about how to interpret them. For instance, we want to keep a distinction between the sequence of symbols $[a \star b]$ and the result of evaluating $f(a, b)$ for a given function $f\colon A^2 \to A$.

With all of that in mind, how do we define what a valid expression is? Since we are treating expressions as syntactic objects, every expression will be a sequence of symbols, i.e. will be an element of Sym_A^*. However, we don't want to include every element of Sym_A^*, because $a[\star \star b][$ should not considered a reasonable expression. The way that we obtain all valid expressions is we start with the elements of A, and build up more complicated expressions by inserting $\star$ between valid expressions, and surrounding with parentheses.

Definition 2.5.1. *Define a binary function $h\colon (Sym_A^*)^2 \to Sym_A^*$ by letting $h(\sigma, \tau)$ be the sequence $[\sigma \star \tau]$. We then have that $(Sym_A^*, A, \{h\})$ is a simple generating system, where we are viewing the elements of A as length 1 sequences in Sym_A. Let $Exp_A = G(Sym_A^*, A, \{h\})$.*

For example, suppose that $A = \{a, b, c\}$. Typical elements of $G(Sym_A^*, A, \{h\})$ are c, $[b \star [a \star c]]$ and $[c \star [[c \star b] \star a]]$. Again, elements of Exp_A are just certain special sequences of symbols, so they do not "mean" anything. In order to attach meaning, we need to have a function $f\colon A^2 \to A$ that will serve as an interpretation of $\star$. Once we have such an f, it seems reasonable that it will provide a way to *evaluate* the elements of Exp_A. That is, with an $f\colon A^2 \to A$ in hand, we should be able to define a function from Exp_A to A, which essentially replaces each occurrence of the symbol $\star$ by an application of f. The natural way to define this function is recursively, but in order to do that, we need to know that the generating system is free.

Proving Freeness

Notice that if we did not use parentheses, i.e. if we defined $h'\colon (Sym_A^*)^2 \to Sym_A^*$ by letting $h'(\sigma, \tau)$ be the sequence $\sigma \star \tau$, then $(Sym_A^*, A, \{h'\})$ would not be free. For example if $A = \{a, b, c\}$, then $a \star b \star c$ would be an element of G, which can be built up in two distinct ways. More formally, we would have $h'(a \star b, c) = h'(a, b \star c)$, so $h' \restriction G^2$ is not be injective.

However, the natural hope is that the inclusion of parentheses forces every element of Exp_A to be generated in a unique way. How can we argue this carefully? Suppose that we have element of Exp_A. Could it be written as both $[\varphi_1 \star \psi_1]$ and $[\varphi_2 \star \psi_2]$ in some nontrivial way (i.e. except in the case where $\varphi_1 = \varphi_2$ and $\psi_1 = \psi_2$)? Since $[\varphi_1 \star \psi_1]$ and $[\varphi_2 \star \psi_2]$ are the same sequence of symbols, either $\varphi_1 = \varphi_2$, or one of the φ_i is a proper initial segment of the other. This is situation that happened in above without parentheses. The sequence $a \star b \star c$ could be decomposed in two ways, and a was a proper initial segment of $a \star b$.

With this in mind, our first goal will be to prove that no element of Exp_A is a proper initial segment of another elements of Exp_A. To accomplish this task, we will employ a simple "weight" function. The idea is as follows. Given a sequence Sym_A of symbols, we scan it from left to right. We give the symbol [a weight of -1, and when we encounter it in a sequence, we think about incurring a small debt that we need to pay off. Analogously, we give the symbol] a weight of 1, and when we encounter it, we think about paying off a small debt. Here is the formal definition.

Definition 2.5.2. *Define $W\colon Sym_A^* \to \mathbb{Z}$ as follows. We begin by defining $w\colon Sym_A \to \mathbb{Z}$ in the following way:*

- $w(a) = 0$ *for all* $a \in A$.
- $w(\star) = 0$.
- $w([) = -1$.
- $w(]) = 1$.

We then define $W : Sym_A^* \to \mathbb{Z}$ *by letting* $W(\lambda) = 0$ *and letting*

$$W(\sigma) = \sum_{i<|\sigma|} w(\sigma(i))$$

for all $\sigma \in Sym_A^* \backslash \{\lambda\}$.

Notice that if $\sigma, \tau \in Sym_A^*$, then we trivially have $W(\sigma\tau) = W(\sigma) + W(\tau)$, since W is just defined as the sums of the weights of the individual symbols. Now the following proposition is intuitively obvious, because any valid expression must have an equal number of left and right parentheses, so every debt will be payed off. Formally, we prove it by induction on the generating system.

Proposition 2.5.3. *If* $\varphi \in Exp_A$*, then* $W(\varphi) = 0$.

Proof. The proof is by induction on φ. In other words, we let $X = \{\varphi \in Exp_A : W(\varphi) = 0\}$, and we prove by induction that $X = Exp_A$. For the base case, notice that for every $a \in A$, we have that $W(a) = 0$. For the inductive step, let $\varphi, \psi \in Exp_A$ be arbitrary with $W(\varphi) = 0 = W(\psi)$. We then have that

$$\begin{aligned} W([\varphi \star \psi]) &= W([) + W(\varphi) + W(\star) + W(\psi) + W(]) \\ &= -1 + 0 + 0 + 0 + 1 \\ &= 0. \end{aligned}$$

By induction, it follows that $X = Exp_A$, concluding the proof. □

Recall that given σ and τ, we use the notation $\sigma \preceq \tau$ to mean that σ is an initial segment of τ, and use $\sigma \prec \tau$ to mean that σ is a proper initial segment of τ, i.e. that $\sigma \preceq \tau$ and $\sigma \neq \tau$.

Proposition 2.5.4. *If* $\varphi \in Exp_A$ *and* $\sigma \prec \varphi$ *with* $\sigma \neq \lambda$*, then* $W(\sigma) \leq -1$.

Proof. Again, the proof is by induction on φ. That is, we let

$$X = \{\varphi \in Exp_A : \text{ For all } \sigma \prec \varphi \text{ with } \sigma \neq \lambda, \text{ we have } W(\sigma) \leq -1\},$$

and we prove by induction that $X = Exp_A$. Notice that the base case is trivial, because given any $a \in A$, there is no $\sigma \neq \lambda$ with $\sigma \prec a$, and hence $a \in X$ vacuously.

For the inductive step, let $\varphi, \psi \in Exp_A$ be arbitrary with $\varphi, \psi \in X$. We prove that $[\varphi \star \psi] \in X$. Let $\sigma \prec [\varphi \star \psi]$ be an arbitrary proper initial segment with $\sigma \neq \lambda$. We handle several cases.

- If σ is [, then $W(\sigma) = -1$.
- If σ is $[\tau$ where $\tau \neq \lambda$ and $\tau \prec \varphi$, then

$$\begin{aligned} W(\sigma) &= -1 + W(\tau) \\ &\leq -1 - 1 && \text{(by induction)} \\ &\leq -1. \end{aligned}$$

- If σ is $[\varphi$ or $[\varphi\star$, then

$$\begin{aligned} W(\sigma) &= -1 + W(\varphi) \\ &= -1 + 0 && \text{(by Proposition 2.5.3)} \\ &= -1. \end{aligned}$$

- If σ is $[\varphi \star \tau$, where $\tau \neq \lambda$ and $\tau \prec \varphi$, then

$$\begin{aligned} W(\sigma) &= -1 + W(\varphi) + W(\tau) \\ &= -1 + 0 + W(\tau) && \text{(by Proposition 2.5.3)} \\ &\leq -1 + 0 - 1 && \text{(by induction)} \\ &\leq -1. \end{aligned}$$

- Otherwise, σ is $[\varphi \star \psi$, and

$$\begin{aligned} W(\sigma) &= -1 + W(\varphi) + W(\psi) \\ &= -1 + 0 + 0 && \text{(by Proposition 2.5.3)} \\ &= -1. \end{aligned}$$

Therefore, in all cases, we have $W(\sigma) \leq -1$. Since σ was an arbitrary proper initial segment of $[\varphi \star \psi]$ with $\sigma \neq \lambda$, we conclude that $[\varphi \star \psi] \in X$.

By induction, it follows that $X = Exp_A$, concluding the proof. □

Corollary 2.5.5. *If $\varphi, \psi \in Exp_A$, then $\varphi \not\prec \psi$.*

Proof. This follows by combining Proposition 2.5.3 and Proposition 2.5.4, along with noting that $\lambda \notin Exp_A$ (which follows by a trivial induction). □

We now have the essential tool that will help us prove freeness.

Theorem 2.5.6. *The generating system $(Sym_A^*, A, \{h\})$ is free.*

Proof. Notice that our simple generating system only has one binary function, so we need only check two things.

- First notice that $\text{range}(h \upharpoonright (Exp_A)^2) \cap A = \emptyset$ because all elements of $\text{range}(h)$ begin with $[$.
- We now show that $\text{range}(h \upharpoonright (Exp_A)^2)$ is injective. Let $\varphi_1, \varphi_2, \psi_1, \psi_2 \in Exp_A$ be arbitrary with $h(\varphi_1, \psi_1) = h(\varphi_2, \psi_2)$. We then have $[\varphi_1 \star \psi_1] = [\varphi_2 \star \psi_2]$, hence $\varphi_1 \star \psi_1 = \varphi_2 \star \psi_2$. Since $\varphi_1 \prec \varphi_2$ and $\varphi_2 \prec \varphi_1$ are both impossible by Corollary 2.5.5, it follows that $\varphi_1 = \varphi_2$. Therefore, $\star\psi_1 = \star\psi_2$, and so $\psi_1 = \psi_2$. It follows that $h \upharpoonright (Exp_A)^2$ is injective.

Combining these two facts, we conclude that the generating system is free. □

The Result

Since we have established freeness, we can define functions on Exp_A recursively. The first such function we define is the "evaluation" function.

Definition 2.5.7. *Let $f : A^2 \to A$. We define a function $Eval_f : Exp_A \to A$ recursively as follows:*

- *$Eval_f(a) = a$ for all $a \in A$.*
- *$Eval_f([\varphi \star \psi]) = f(Eval_f(\varphi), Eval_f(\psi))$ for all $\varphi, \psi \in Exp_A$.*

Formally, here is how we use freeness to justify the definition. Let $\alpha : A \to A$ be the identity map, and let $g_h : (Sym_A^* \times A)^2 \to A$ be the function defined by letting $g_h(\sigma, a, \tau, b) = f(a, b)$. By Theorem 2.4.5, there is a unique function $Eval_f : Exp_A \to A$ with the following two properties:

1. $Eval_f(a) = \alpha(a)$ for all $a \in A$.

2. $Eval_f(h(\varphi,\psi)) = g_h(\varphi, Eval_f(\varphi), \psi, Eval_f(\psi))$ for all $\varphi,\psi \in Exp_A$.

Unraveling definitions, this is exactly what we wrote above. In the future, we will typically avoid this level of formality, and just define recursive functions as in the above definition.

Recall that our goal is to prove the following: If $f\colon A^2 \to A$ is associative, i.e. $f(a,f(b,c)) = f(f(a,b),c)$ for all $a,b,c \in A$, then any "grouping" of terms which preserves the ordering of the elements inside the grouping gives the same value. In order to define "preserves the ordering of the elements" carefully, we now introduce a function that eliminates all parentheses and occurrences of $\star$ in an element of Exp_A. In other words, it produces the sequence of elements from A within the given expression, in order of their occurrence. Since the function destroys characters, we'll call it D. Notice that D is also defined recursively.

Definition 2.5.8. *Define a function $D\colon Exp_A \to A^*$ recursively as follows:*

- $D(a) = a$ *for all* $a \in A$.
- $D([\varphi\star\psi]) = D(\varphi)D(\psi)$ *for all* $\varphi,\psi \in Exp_A$, *where* $D(\varphi)D(\psi)$ *is just the concatenation of the sequences* $D(\varphi)$ *and* $D(\psi)$.

With these definitions in hand, we can now precisely state our theorem.

Theorem 2.5.9. *Suppose that $f\colon A^2 \to A$ is associative, i.e. $f(a,f(b,c)) = f(f(a,b),c)$ for all $a,b,c \in A$. For all $\varphi,\psi \in Exp_A$ with $D(\varphi) = D(\psi)$, we have $Eval_f(\varphi) = Eval_f(\psi)$.*

In order to prove our theorem, we need to a way to take a sequence $\sigma \in A^*$ of elements of A, and provide a canonical element $\varphi \in Exp_A$ with $D(\varphi) = \sigma$ that we can evaluate. The most natural way to do this is to pick a side to group terms on. We'll choose to "associate to the right", so that the sequence *cabc* will produce $[c\star[a\star[b\star c]]]$. The definition is intuitively clear, but to make it more precise, we define this grouping recursively. We could define it recursively on the length of an element of A^*, but its more elegant to use the simple generating system $(A^*, A, \{h_a : a \in A\})$ where $h_a\colon A^* \to A^*$ is defined by $h_a(\sigma) = a\sigma$. As shown in Example 2.4.4, we know that $(A^*, A, \{h_a : a \in A\})$ is free and we have that $G = A^*\backslash\{\lambda\}$, which justifies the following recursive definition.

Definition 2.5.10. *We define $R\colon A^*\backslash\{\lambda\} \to Sym^*_A$ recursively by letting $R(a) = a$ for all $a \in A$, and letting $R(a\sigma) = [a \star R(\sigma)]$ for all $a \in A$ and all $\sigma \in A^*\backslash\{\lambda\}$.*

The following result can be proven by a simple induction on the generating system $(A^*, A, \{h_a : a \in A\})$.

Proposition 2.5.11. *For all $\sigma \in A^*\backslash\{\lambda\}$, we have $R(\sigma) \in Exp_A$.*

Now in order to prove Theorem 2.5.9, we will show that $Eval_f(\varphi) = Eval_f(R(D(\varphi)))$ for all $\varphi \in Exp_A$, i.e. that we can take any $\varphi \in Exp_A$, rip it apart so that we see the elements of A in order, and then associate to the right, without affecting the result of the evaluation. We first need the following lemma.

Lemma 2.5.12. *$Eval_f([R(\sigma) \star R(\tau)]) = Eval_f(R(\sigma\tau))$ for all $\sigma,\tau \in A^*\backslash\{\lambda\}$.*

Proof. Let $\tau \in A^*\backslash\{\lambda\}$ be arbitrary. We prove that the statement is true for this fixed τ by induction on $A^*\backslash\{\lambda\}$. That is, we let

$$X = \{\sigma \in A^*\backslash\{\lambda\} : Eval_f([R(\sigma) \star R(\tau)]) = Eval_f(R(\sigma\tau))\},$$

and prove by induction on $(A^*, A, \{h_a : a \in A\})$ that $X = A^*\backslash\{\lambda\}$. Suppose first that $a \in A$. We then have

$$\begin{aligned} Eval_f([R(a) \star R(\tau)]) &= Eval_f([a \star R(\tau)]) && \text{(by definition of } R\text{)} \\ &= Eval_f(R(a\tau)) && \text{(by definition of } R\text{),} \end{aligned}$$

so $a \in X$. Suppose now that $\sigma \in X$ and that $a \in A$. We show that $a\sigma \in X$. We have

$$\begin{aligned}
Eval_f([R(a\sigma) \star R(\tau)]) &= Eval_f([[a \star R(\sigma)] \star R(\tau)]) && \text{(by definition of } R\text{)} \\
&= f(Eval_f([a \star R(\sigma)]), Eval_f(R(\tau))) && \text{(by definition of } Eval_f\text{)} \\
&= f(f(a, Eval_f(R(\sigma))), Eval_f(R(\tau))) && \text{(by definition of } Eval_f\text{, using } Eval_f(a) = a\text{)} \\
&= f(a, f(Eval_f(R(\sigma)), Eval_f(R(\tau)))) && \text{(since } f \text{ is associative)} \\
&= f(a, Eval_f([R(\sigma) \star R(\tau)])) && \text{(by definition of } Eval_f\text{)} \\
&= f(a, Eval_f(R(\sigma\tau))) && \text{(since } \sigma \in X\text{)} \\
&= Eval_f([a \star R(\sigma\tau)]) && \text{(by definition of } Eval_f\text{, using } Eval_f(a) = a\text{)} \\
&= Eval_f(R(a\sigma\tau)) && \text{(by definition of } R\text{),}
\end{aligned}$$

so $a\sigma \in X$. The result follows by induction. □

We can now prove our key lemma.

Lemma 2.5.13. *$Eval_f(\varphi) = Eval_f(R(D(\varphi)))$ for all $\varphi \in Exp_A$.*

Proof. By induction on Exp_A. If $a \in A$, this is trivial because $R(D(a)) = R(a) = a$. Suppose that $\varphi, \psi \in Exp_A$ and the statement is true for φ and ψ. We then have

$$\begin{aligned}
Eval_f([\varphi \star \psi]) &= f(Eval_f(\varphi), Eval_f(\psi)) && \text{(by definition of } Eval_f\text{)} \\
&= f(Eval_f(R(D(\varphi))), Eval_f(R(D(\psi)))) && \text{(by induction)} \\
&= Eval_f([R(D(\varphi)) \star R(D(\psi))]) && \text{(by definition of } Eval_f\text{)} \\
&= Eval_f(R(D(\varphi)D(\psi))) && \text{(by Lemma 2.5.12)} \\
&= Eval_f(R(D([\varphi \star \psi]))) && \text{(by definition of } D\text{).}
\end{aligned}$$

□

Finally, we can finish the proof of our theorem.

Proof of Theorem 2.5.9. Let $\varphi, \psi \in Exp_A$ be arbitrary such that $D(\varphi) = D(\psi)$. We have

$$\begin{aligned}
Eval_f(\varphi) &= Eval_f(R(D(\varphi))) && \text{(by Lemma 2.5.13)} \\
&= Eval_f(R(D(\psi))) && \text{(since } D(\varphi) = D(\psi)\text{)} \\
&= Eval_f(\psi) && \text{(by Lemma 2.5.13).}
\end{aligned}$$

□

It's certainly reasonable to ask if the amount of formality and rigor that we used to prove this theorem was worth it. After all, the result was intuitive and reasonably obvious. These concerns are certainly valid, but working through all of the details in this simple setting will ease the transition to more complicated arguments.

An Alternate Syntax - Polish Notation

It is standard mathematical practice to place binary operations like $\star$ between two elements (called "infix notation") to signify the application of a binary function, and throughout this section we have followed that tradition in building up valid expressions. However, the price that we have pay is that we needed to use parentheses to avoid ambiguity, i.e. to provide freeness. As mentioned, without parentheses, it is not clear

how to parse $a \star b \star c$. Should it be $[[a \star b] \star c]$ or $[a \star [b \star c]]$? If the underlying function f is not associative, then the distinction really matters.

We can of course move the operation to the front and write $\star[a, b]$ instead of $[a \star b]$ similar to how we might write $f(x, y)$ for a function of two variables. At first sight, this looks even worse because now we introduced a comma. However, it turns out that we can eliminate all of the extra symbols. That is, we simply write $\star ab$ without any additional punctuation ,and build further expressions up in this way, then we avoid any ambiguity. This syntactic approach is called "Polish notation". For example, we have the following translations in Polish notation.

- $[[a \star b] \star c] \Longrightarrow \star\star abc$
- $[a \star [b \star c]] \Longrightarrow \star a \star bc$.
- $[[a \star b] \star [c \star d]] \Longrightarrow \star\star ab \star cd$.

We now go about proving that every expression in Polish notation is built up in a unique way. That is, we prove that the corresponding generating system is free. For this section, let A be a set not containing the symbol $\star$ and let $Sym_A = A \cup \{\star\}$. That is, we no longer include parentheses in Sym_A.

Definition 2.5.14. *Define a binary function $h\colon (Sym_A^*)^2 \to Sym_A^*$ by letting $h(\sigma, \tau)$ be the sequence $\star\sigma\tau$. We then have that $(Sym_A^*, A, \{h\})$ is a simple generating system, where we are viewing the elements of A as length 1 sequences in Sym_A^*. Let $PolishExp_A = G(Sym_A^*, A, \{h\})$.*

In order to prove that $(Sym_A^*, A, \{h\})$ is free, we follow the structure of our previous argument. We start by defining a weight function, but here it is less obvious how to proceed. Unlike the previous case, where parentheses carried all of the weight and the elements of A were neutral, we now can only have the elements of A to signify when we have paid off a debt. As a result, instead of giving elements of A a weight of 0, we will give them weight 1. This leads to the following definition.

Definition 2.5.15. *Define $W\colon Sym_A^* \to \mathbb{Z}$ as follows. We begin by defining $w\colon Sym_A \to \mathbb{Z}$ in the following way:*

- *$w(a) = 1$ for all $a \in A$.*
- *$w(\star) = -1$.*

We then define $W\colon Sym_A^* \to \mathbb{Z}$ by letting $W(\lambda) = 0$ and letting

$$W(\sigma) = \sum_{i<|\sigma|} w(\sigma(i))$$

for all $\sigma \in Sym_A^* \backslash \{\lambda\}$.

Notice again that if $\sigma, \tau \in Sym_A^*$, then we trivially have $W(\sigma\tau) = W(\sigma) + W(\tau)$. Now when we scan an element of Sym_A^* from left to right, we invoke a debt of -1 when we run across a $\star$, and end up paying back 2 when we encounter elements of A. As a result, valid expressions now have weight 1 instead of weight 0.

Proposition 2.5.16. *If $\varphi \in PolishExp_A$, then $W(\varphi) = 1$.*

Proof. The proof is by induction on φ. Notice that for every $a \in A$, we have that $W(a) = 1$ by definition. Let $\varphi, \psi \in PolishExp_A$ be arbitrary with $W(\varphi) = 1 = W(\psi)$. We then have that

$$\begin{aligned} W(\star\varphi\psi) &= W(\star) + W(\varphi) + W(\psi) \\ &= W(\varphi) \\ &= 1. \end{aligned}$$

The result follows by induction. □

We now show that proper initial segments of valid expressions have smaller weight. In this case, we do not have to treat λ differently.

Proposition 2.5.17. *If $\varphi \in PolishExp_A$ and $\sigma \prec \varphi$, then $W(\sigma) \leq 0$.*

Proof. The proof is by induction on φ. Again the base case is trivial, because given any $a \in A$, the only $\sigma \prec a$ is $\sigma = \lambda$, and we have $W(\lambda) = 0$. For the inductive step, assume that $\varphi, \psi \in PolishExp_A$ and that the statement is true for φ and ψ. We prove that the statement is true for $\star\varphi\psi$. Let $\sigma \prec \star\varphi\psi$ be arbitrary. We have several cases.

- If $\sigma = \lambda$, then $W(\sigma) = 0$.
- If σ is $\star\tau$ for some $\tau \prec \varphi$, then

$$\begin{aligned} W(\sigma) &= W(\star) + W(\tau) \\ &\leq -1 + 0 && \text{(by induction)} \\ &\leq -1 \\ &\leq 0. \end{aligned}$$

- Otherwise, σ is $\star\varphi\tau$ for some $\tau \prec \psi$, in which case

$$\begin{aligned} W(\sigma) &= W(\star) + W(\varphi) + W(\tau) \\ &= -1 + 1 + W(\tau) && \text{(by Proposition 2.5.16)} \\ &\leq -1 + 1 + 0 && \text{(by induction)} \\ &\leq 0. \end{aligned}$$

Thus, the statement is true for $\star\varphi\psi$. □

Corollary 2.5.18. *If $\varphi, \psi \in PolishExp_A$, then $\varphi \not\prec \psi$.*

Proof. This follows by combining Proposition 2.5.16 and Proposition 2.5.17. □

Theorem 2.5.19. *The generating system $(Sym_A^*, A, \{h\})$ is free.*

Proof. Notice that our simple generating system only has one binary function, so we need only check two things.

- First notice that $\text{range}(h \upharpoonright (PolishExp_A)^2) \cap A = \emptyset$ because all elements of $\text{range}(h)$ begin with $\star$.
- We now show that $\text{range}(h \upharpoonright (PolishExp_A)^2)$ is injective. Suppose that $\varphi_1, \varphi_2, \psi_1, \psi_2 \in PolishExp_A$ and that $h(\varphi_1, \psi_1) = h(\varphi_2, \psi_2)$. We then have $\star\varphi_1\psi_1 = \star\varphi_2\psi_2$, hence $\varphi_1\psi_1 = \varphi_2\psi_2$. Since $\varphi_1 \prec \varphi_2$ and $\varphi_2 \prec \varphi_1$ are both impossible by Corollary 2.5.18, it follows that $\varphi_1 = \varphi_2$. Therefore, $\psi_1 = \psi_2$. It follows that $h \upharpoonright (PolishExp_A)^2$ is injective.

Combining these two facts, we conclude that the generating system is free. □

If we wanted, we could recursively define an evaluation function (given an $f\colon A^2 \to A$), and prove analogous results. However, now that we have become acquainted with Polish notation, we can move on to our study of logic.

2.6 Exercises

1. Consider the following simple generating system. Let $A = \{1, 2, 3, 4, 5, 6, 7\}$, $B = \{5\}$ and $\mathcal{H} = \{h_1, h_2\}$, where $h_1 \colon A \to A$ is given by

$$h_1(1) = 3 \quad h_1(2) = 1 \quad h_1(3) = 3 \quad h_1(4) = 7 \quad h_1(5) = 5 \quad h_1(6) = 1 \quad h_1(7) = 4$$

and $h_2 \colon A^2 \to A$ is given by

	1	2	3	4	5	6	7
1	3	6	5	6	2	2	1
2	7	2	1	1	2	1	3
3	5	6	5	1	2	2	3
4	1	4	4	4	4	4	7
5	4	5	2	5	7	3	4
6	1	7	5	1	2	1	2
7	7	6	1	5	5	1	5

Interpret the diagram as follows. If $m, n \in A$, to calculate the value of $h_2(m, n)$, go to row m and column n. For example, $h_2(1, 2) = 6$.

   (a) Determine V_3 and W_3 explicitly. Justify your answers.

   (b) Determine G explicitly. Justify your answer.

2. Let A be an arbitrary group, and let B be an arbitrary nonempty subset of A. Let $\mathcal{H} = \{h_1, h_2\}$ where $h_1 \colon A \to A$ is the inverse function and $h_2 \colon A^2 \to A$ is the group operation. Show that the simple generating system $(A, B, \mathcal{H})$ is not free.

3. Let $A = \mathbb{R}^+ = \{x \in \mathbb{R} : x > 0\}$, $B_1 = \{\sqrt{2}\}$, $B_2 = \{\sqrt{2}, 16\}$, and $\mathcal{H} = \{h\}$ where $h \colon \mathbb{R}^+ \to \mathbb{R}^+$ is given by $h(x) = x^2$.

   (a) Describe $G(A, B_1, \mathcal{H})$ and $G(A, B_2, \mathcal{H})$ explicitly.

   (b) Show that $(A, B_1, \mathcal{H})$ is free, but $(A, B_2, \mathcal{H})$ is not free.

   (c) Define $\alpha \colon B_2 \to \mathbb{R}$ by $\alpha(\sqrt{2}) = 0$ and $\alpha(16) = \frac{7}{2}$. Define $g_h \colon \mathbb{R}^+ \times \mathbb{R} \to \mathbb{R}$ by letting $g_h(a, x) = \log_2 a + x$. Show that there exists a function $f \colon G(A, B_2, \mathcal{H}) \to \mathbb{R}$ with the following properties:
      - $f(b) = \alpha(b)$ for all $b \in B_2$.
      - $f(h(a)) = g_h(a, f(a))$ for all $a \in G(A, B_2, \mathcal{H})$.

   Thus, we can define a function that satisfies the recursive equations in this case despite the lack of freeness.

4. Let $A = \mathbb{N}^+$, $B = \{7, 13\}$, and $\mathcal{H} = \{h_1, h_2\}$ where $h_1 \colon A \to A$ is given by $h_1(n) = 20n + 1$ and $h_2 \colon A^2 \to A$ is given by $h_2(n, m) = 2^n(2m + 1)$. Show that $(A, B, \mathcal{H})$ is free.

5. Let $\overline{\mathbb{Q}}$ be the set of all algebraic numbers. That is, $\overline{\mathbb{Q}}$ is the set of all $z \in \mathbb{C}$ such that there exists a nonzero polynomial $p(x) \in \mathbb{Q}[x]$ with rational coefficients having z as a root. Show that $\overline{\mathbb{Q}}$ is countable.

6. Let $(A, B, \mathcal{H})$ be a (not necessarily simple) generating system. Assume that B is countable, that $\mathcal{H}$ is countable (that is, there are countably many functions in $\mathcal{H}$), and that for each $h \in \mathcal{H}_k$ and $a_1, a_2, \ldots, a_k \in A$, the set $h(a_1, a_2, \ldots, a_k)$ is countable. Show that G is countable.

7. (**) Suppose that $(A, B, \mathcal{H})$ is a simple generating system that is not free. Show that there exists a set X and functions $\alpha\colon B \to X$ and $g_h\colon (A \times X)^k \to X$ for each $h \in \mathcal{H}_k$, such that there is no function $f\colon G \to X$ satisfying the following:

   - $f(b) = \alpha(b)$ for all $b \in B$.
   - $f(h(a_1, a_2, \ldots, a_k)) = g_h(a_1, f(a_1), a_2, f(a_2), \ldots, a_k, f(a_k))$ for all $h \in \mathcal{H}_k$ and all $a_1, a_2, \ldots, a_k \in G$.

Chapter 3

Propositional Logic

3.1 The Syntax of Propositional Logic

We now embark on a careful study of Propositional Logic. As described in Chapter 1, in this setting, we start with an arbitrary set P, which we think of as our collection of primitive statements. From here, we build up more complicated statements by repeatedly applying connectives. As in Section 2.5, we have multiple syntactic approaches that we can follow. We develop both here.

Standard Syntax

We start with the more human readable syntax that uses infix notation for binary connectives, and hence must employ parentheses in order to avoid ambiguity. In Section 2.5, we used square brackets to distinguish the formal syntactic constructs from their normal mathematical use. Since we have some experience now, we will forgo that pedantic distinction here in order to have more natural looking objects.

Definition 3.1.1. *Let P be a nonempty set not containing the symbols $(,), \neg, \wedge, \vee$, and $\rightarrow$, and define $Sym_P = P \cup \{(,), \neg, \wedge, \vee, \rightarrow\}$. Define a unary function $h_\neg$ and binary functions $h_\wedge, h_\vee$, and $h_\rightarrow$ on Sym_P^* as follows:*

$$\begin{aligned} h_\neg(\sigma) &= (\neg\sigma) \\ h_\wedge(\sigma, \tau) &= (\sigma \wedge \tau) \\ h_\vee(\sigma, \tau) &= (\sigma \vee \tau) \\ h_\rightarrow(\sigma, \tau) &= (\sigma \rightarrow \tau). \end{aligned}$$

We then let $Form_P = G(Sym_P^*, P, \mathcal{H})$ where $\mathcal{H} = \{h_\neg, h_\wedge, h_\vee, h_\rightarrow\}$

In other words, we generate more complicated statements by starting with the elements of P and applying the functions that introduce connectives. We call the results syntactic objects *formulas*. We now argue the generating system is free by following the outline in Section 2.5.

Definition 3.1.2. *Define $W\colon Sym_P^* \rightarrow \mathbb{Z}$ as follows. We begin by defining $w\colon Sym_P \rightarrow \mathbb{Z}$ in the following way:*

- $w(\mathsf{A}) = 0$ *for all* $\mathsf{A} \in P$.
- $w(\diamond) = 0$ *for all* $\diamond \in \{\neg, \wedge, \vee, \rightarrow\}$.
- $w(() = -1$.

- $w()) = 1$.

We then define $W\colon Sym_P^* \to \mathbb{Z}$ *by letting* $W(\lambda) = 0$ *and letting*

$$W(\sigma) = \sum_{i<|\sigma|} w(\sigma(i))$$

for all $\sigma \in Sym_P^* \backslash \{\lambda\}$.

With this weight function in hand, the proof that the generating system is free proceeds in the same way as the example in Section 2.5.

Proposition 3.1.3. *We have the following:*

1. *If* $\varphi \in Form_P$, *then* $W(\varphi) = 0$.
2. *If* $\varphi \in Form_P$ *and* $\sigma \prec \varphi$ *with* $\sigma \neq \lambda$, *then* $W(\sigma) \leq -1$.
3. *If* $\varphi \in Form_P$, *then* $\varphi \neq \lambda$, *and either* φ *is an element of* P, *or* φ *begins with* (.

Proof. These all follow using straightforward inductive arguments (with multiple inductive steps, since we now have 4 generating functions), the first two of which are completely analogous to Proposition 2.5.3 and Proposition 2.5.4. □

Corollary 3.1.4. *If* $\varphi, \psi \in Form_P$, *then* $\varphi \not\prec \psi$.

Proof. First notice that $\varphi \neq \lambda$ by part (3) of Proposition 3.1.3. Now apply parts (1) and (2) of Proposition 3.1.3. □

Theorem 3.1.5. *The generating system* $(Sym_P^*, P, \mathcal{H})$ *is free.*

Proof. We have to check the three properties.

- First notice that $\text{range}(h_\neg \restriction Form_P) \cap P = \emptyset$ because all elements of $\text{range}(h_\neg)$ begin with (. Similarly, for any $\diamond \in \{\wedge, \vee, \to\}$, we have $\text{range}(h_\diamond \restriction Form_P^2) \cap P = \emptyset$ since all elements of $\text{range}(h_\diamond)$ begin with (.

- We next need to check that each element of $\mathcal{H}$ is injective, when restricted to formulas.

  Let $\varphi, \psi \in Form_P$ be arbitrary with $h_\neg(\varphi) = h_\neg(\psi)$. We then have $(\neg\varphi) = (\neg\psi)$, hence $\varphi = \psi$. Therefore, $h_\neg \restriction Form_P$ is injective.

  Suppose $\diamond \in \{\wedge, \vee, \to\}$. Let $\varphi_1, \varphi_2, \psi_1, \psi_2 \in Form_P$ be arbitrary with $h_\diamond(\varphi_1, \psi_1) = h_\diamond(\varphi_2, \psi_2)$. We then have $(\varphi_1 \diamond \psi_1) = (\varphi_2 \diamond \psi_2)$, hence $\varphi_1 \diamond \psi_1 = \varphi_2 \diamond \psi_2$. Since $\varphi_1 \prec \varphi_2$ and $\varphi_2 \prec \varphi_1$ are both impossible by Corollary 3.1.4, it follows that $\varphi_1 = \varphi_2$. Therefore, $\diamond\psi_1 = \diamond\psi_2$, and so $\psi_1 = \psi_2$. It follows that $h_\diamond \restriction Form_P^2$ is injective.

- Finally, we must show that two distinct elements of $\mathcal{H}$ have disjoint ranges, when restricted to formulas.

  Suppose $\diamond \in \{\wedge, \vee, \to\}$. Let $\varphi, \psi_1, \psi_2 \in Form_P$ be arbitrary with $h_\neg(\varphi) = h_\diamond(\psi_1, \psi_2)$. We then have $(\neg\varphi) = (\psi_1 \diamond \psi_2)$, hence $\neg\varphi = \psi_1 \diamond \psi_2$, contradicting the fact that no element of $Form_P$ begins with $\neg$ (by part (3) of Proposition 3.1.3). Therefore, $\text{range}(h_\neg \restriction Form_P) \cap \text{range}(h_\diamond \restriction Form_P^2) = \emptyset$.

  Suppose $\diamond_1, \diamond_2 \in \{\wedge, \vee, \to\}$ with $\diamond_1 \neq \diamond_2$. Let $\varphi_1, \varphi_2, \psi_1, \psi_2 \in Form_P$ be arbitrary with $h_{\diamond_1}(\varphi_1, \psi_1) = h_{\diamond_2}(\varphi_2, \psi_2)$. We then have $(\varphi_1 \diamond_1 \psi_1) = (\varphi_2 \diamond_2 \psi_2)$, hence $\varphi_1 \diamond_1 \psi_1 = \varphi_2 \diamond_2 \psi_2$. Since $\varphi_1 \prec \varphi_2$ and $\varphi_2 \prec \varphi_1$ are both impossible by Corollary 3.1.4, it follows that $\varphi_1 = \varphi_2$. Therefore, $\diamond_1 = \diamond_2$, a contradiction. It follows that $\text{range}(h_{\diamond_1} \restriction Form_P^2) \cap \text{range}(h_{\diamond_2} \restriction Form_P^2) = \emptyset$.

□

Polish Notation

Definition 3.1.6. *Let P be a set not containing the symbols $\neg, \wedge, \vee$, and $\rightarrow$, and define $Sym_P = P \cup \{\neg, \wedge, \vee, \rightarrow\}$. Define a unary function $h_\neg$ and binary functions $h_\wedge, h_\vee$, and $h_\rightarrow$ on Sym_P^* as follows:*

$$\begin{aligned} h_\neg(\sigma) &= \neg\sigma \\ h_\wedge(\sigma, \tau) &= \wedge\sigma\tau \\ h_\vee(\sigma, \tau) &= \vee\sigma\tau \\ h_\rightarrow(\sigma, \tau) &= \rightarrow \sigma\tau. \end{aligned}$$

We then let $Form_P = G(Sym_P^*, P, \mathcal{H})$ where $\mathcal{H} = \{h_\neg, h_\wedge, h_\vee, h_\rightarrow\}$.

Definition 3.1.7. *Define $W\colon Sym_P^* \rightarrow \mathbb{Z}$ as follows. We begin by defining $w\colon Sym_P \rightarrow \mathbb{Z}$ in the following way:*

- *$w(A) = 1$ for all $A \in P$.*
- *$w(\neg) = 0$.*
- *$w(\diamond) = -1$ for all $\diamond \in \{\wedge, \vee, \rightarrow\}$.*

We then define $W\colon Sym_P^* \rightarrow \mathbb{Z}$ by letting $W(\lambda) = 0$ and letting

$$W(\sigma) = \sum_{i<|\sigma|} w(\sigma(i))$$

for all $\sigma \in Sym_P^* \backslash \{\lambda\}$.

Proposition 3.1.8. *If $\varphi \in Form_P$, then $W(\varphi) = 1$.*

Proof. The proof is by induction on φ. Notice that for every $A \in P$, we have that $W(A) = 1$. Suppose that $\varphi \in Form_P$ is such that $W(\varphi) = 1$. We then have that

$$\begin{aligned} W(\neg\varphi) &= 0 + W(\varphi) \\ &= W(\varphi) \\ &= 1. \end{aligned}$$

Suppose now that $\varphi, \psi \in Form_P$ are such that $W(\varphi) = 1 = W(\psi)$, and $\diamond \in \{\wedge, \vee, \rightarrow\}$. We then have that

$$\begin{aligned} W(\diamond\varphi\psi) &= -1 + W(\varphi) + W(\psi) \\ &= -1 + 1 + 1 \\ &= 1. \end{aligned}$$

The result follows by induction. □

Proposition 3.1.9. *If $\varphi \in Form_P$ and $\sigma \prec \varphi$, then $W(\sigma) \leq 0$.*

Proof. The proof is by induction on φ. For every $A \in P$, this is trivial because the only $\sigma \prec A$ is $\sigma = \lambda$, and we have $W(\lambda) = 0$.

Suppose that $\varphi \in Form_P$ and the statement is true for φ. We prove that the statement is true for $\neg\varphi$. Suppose that $\sigma \prec \neg\varphi$. If $\sigma = \lambda$, then $W(\sigma) = 0$. Otherwise, σ is $\neg\tau$ for some $\tau \prec \varphi$, in which case

$$\begin{aligned} W(\sigma) &= 0 + W(\tau) \\ &\leq 0 + 0 && \text{(by induction)} \\ &\leq 0. \end{aligned}$$

Thus, the statement is true for $\neg\varphi$.

Suppose that $\varphi, \psi \in Form_P$ and the statement is true for φ and ψ. Let $\diamond \in \{\wedge, \vee, \rightarrow\}$. We prove that the statement is true for $\diamond\varphi\psi$. Suppose that $\sigma \prec \diamond\varphi\psi$. If $\sigma = \lambda$, then $W(\sigma) = 0$. If σ is $\diamond\tau$ for some $\tau \prec \varphi$, then

$$\begin{aligned} W(\sigma) &= -1 + W(\tau) \\ &\leq -1 + 0 && \text{(by induction)} \\ &\leq -1 \\ &\leq 0. \end{aligned}$$

Otherwise, σ is $\diamond\varphi\tau$ for some $\tau \prec \psi$, in which case

$$\begin{aligned} W(\sigma) &= -1 + W(\varphi) + W(\tau) \\ &= -1 + 1 + W(\tau) && \text{(by Proposition 3.1.8)} \\ &\leq -1 + 1 + 0 && \text{(by induction)} \\ &\leq 0. \end{aligned}$$

Thus, the statement is true for $\diamond\varphi\psi$. □

Corollary 3.1.10. *If $\varphi, \psi \in Form_P$, then $\varphi \not\prec \psi$.*

Proof. This follows by combining Proposition 3.1.8 and Proposition 3.1.9. □

Theorem 3.1.11. *The generating system $(Sym_P^*, P, \mathcal{H})$ is free.*

Proof. We have to check the three properties.

- First notice that $\text{range}(h_\neg \restriction Form_P) \cap P = \emptyset$ because all elements of $\text{range}(h_\neg)$ begin with $\neg$. Similarly, for any $\diamond \in \{\wedge, \vee, \rightarrow\}$, we have $\text{range}(h_\diamond \restriction Form_P^2) \cap P = \emptyset$ since all elements of $\text{range}(h_\diamond)$ begin with $\diamond$.

- We next need to check that each element of $\mathcal{H}$ is injective, when restricted to formulas.

  Let $\varphi, \psi \in Form_P$ be arbitrary with $h_\neg(\varphi) = h_\neg(\psi)$. We then have $\neg\varphi = \neg\psi$, hence $\varphi = \psi$. Therefore, $h_\neg \restriction Form_P$ is injective.

  Suppose $\diamond \in \{\wedge, \vee, \rightarrow\}$. Let $\varphi_1, \varphi_2, \psi_1, \psi_2 \in Form_P$ be arbitrary with $h_\diamond(\varphi_1, \psi_1) = h_\diamond(\varphi_2, \psi_2)$. We then have $\diamond\varphi_1\psi_1 = \diamond\varphi_2\psi_2$, hence $\varphi_1\psi_1 = \varphi_2\psi_2$. Since $\varphi_1 \prec \varphi_2$ and $\varphi_2 \prec \varphi_1$ are both impossible by Corollary 3.1.10, it follows that $\varphi_1 = \varphi_2$. Therefore, $\psi_1 = \psi_2$. Therefore, $h_\diamond \restriction Form_P^2$ is injective.

- Finally, we must show that two distinct elements of $\mathcal{H}$ have disjoint ranges, when restricted to formulas.

  Suppose $\diamond \in \{\wedge, \vee, \rightarrow\}$. We have $\text{range}(h_\neg \restriction Form_P) \cap \text{range}(h_\diamond \restriction Form_P^2) = \emptyset$ because all elements of $\text{range}(h_\neg)$ begin with $\neg$ and all elements of $\text{range}(h_\diamond)$ begin with $\diamond$.

  Suppose $\diamond_1, \diamond_2 \in \{\wedge, \vee, \rightarrow\}$ with $\diamond_1 \neq \diamond_2$. We have $\text{range}(h_{\diamond_1} \restriction Form_P^2) \cap \text{range}(h_{\diamond_2} \restriction Form_P^2) = \emptyset$ because all elements of $\text{range}(h_{\diamond_1})$ begin with $\diamond_1$ and all elements of $\text{range}(h_{\diamond_2})$ begin with $\diamond_2$.

□

Official Syntax and Our Abuses of It

Since we should probably fix an official syntax, we'll choose to use Polish notation. The ability to use fewer symbols is more elegant, and we'll find that it is more natural to generalize to later context (when we talk about the possibility of adding new connectives, and when we get to first-order logic). However, as with many official definitions in mathematics, we'll ignore and abuse this convention constantly in the interest of readability. For example, we'll often write things in standard syntax or in more abbreviated forms. For example, we'll write $\mathsf{A} \wedge \mathsf{B}$ instead of either $\wedge\mathsf{AB}$ or $(\mathsf{A} \wedge \mathsf{B})$. We'll also write something like

$$\mathsf{A}_1 \wedge \mathsf{A}_2 \wedge \cdots \wedge \mathsf{A}_{n-1} \wedge \mathsf{A}_n$$

or

$$\bigwedge_{i=1}^{n} \mathsf{A}_i$$

in place of $(\mathsf{A}_1 \wedge (\mathsf{A}_2 \wedge (\cdots (\mathsf{A}_{n-1} \wedge \mathsf{A}_n) \cdots)))$ in standard syntax or $\wedge\mathsf{A}_1 \wedge \mathsf{A}_2 \cdots \wedge \mathsf{A}_{n-1}\mathsf{A}_n$ in Polish notation. Notice that each of these can be defined formally by using a variant of the function R defined in Section 2.5. In general, when we string together multiple applications of an operation (such as $\wedge$) in order, we always associate to the right.

When it comes to mixing symbols, we'll follow conventions about "binding" similar to how we think of $\cdot$ as more binding than $+$ (so that $3 \cdot 5 + 2$ is read as $(3 \cdot 5) + 2$). We think of $\neg$ as the most binding, so we read $\neg\mathsf{A} \wedge \mathsf{B}$ as $((\neg\mathsf{A}) \wedge \mathsf{B})$. After that, we consider $\wedge$ and $\vee$ as the next most binding, and $\to$ has the least binding. We'll insert parentheses when we wish to override this binding. For example, $\mathsf{A} \wedge \neg\mathsf{B} \to \mathsf{C} \vee \mathsf{D}$ is really $((\mathsf{A} \wedge (\neg\mathsf{B})) \to (\mathsf{C} \vee \mathsf{D}))$ while $\mathsf{A} \wedge (\neg\mathsf{B} \to \mathsf{C} \vee \mathsf{D})$ is really $(\mathsf{A} \wedge ((\neg\mathsf{B}) \to (\mathsf{C} \vee \mathsf{D})))$.

Recursive Definitions

Since we've shown that our generating system is free, we can define functions recursively. Now it is possible to define some of functions directly, without appealing to recursion. In such cases, you may wonder why we bother. Since our only powerful way to prove things about the set $Form_P$ is by induction, and definitions of functions by recursion are well-suited to induction, it's simply the easiest way to proceed.

Our first function takes as input a formula φ, and outputs the set of element of P that occur within the formula φ.

Definition 3.1.12. *Given a set P, we define a function $OccurProp\colon Form_P \to \mathcal{P}(P)$ recursively as follows:*

- $OccurProp(\mathsf{A}) = \{\mathsf{A}\}$ *for all* $\mathsf{A} \in P$.
- $OccurProp(\neg\varphi) = OccurProp(\varphi)$.
- $OccurProp(\diamond\varphi\psi) = OccurProp(\varphi) \cup OccurProp(\psi)$ *for each* $\diamond \in \{\wedge, \vee, \to\}$.

In order to make the definition precise, we're starting with functions $\alpha\colon P \to \mathcal{P}(P)$, $g_{h_\neg}\colon Sym_P^* \times \mathcal{P}(P) \to \mathcal{P}(P)$ and $g_{h_\diamond}\colon (Sym_P^* \times \mathcal{P}(P))^2 \to \mathcal{P}(P)$ for each $\diamond \in \{\wedge, \vee, \to\}$ defined as follows:

- $\alpha(\mathsf{A}) = \{\mathsf{A}\}$ for all $\mathsf{A} \in P$.
- $g_{h_\neg}(\sigma, Z) = Z$.
- $g_{h_\diamond}(\sigma_1, Z_1, \sigma_2, Z_2) = Z_1 \cup Z_2$ for each $\diamond \in \{\wedge, \vee, \to\}$.

We are then appealing to Theorem 2.4.5, which tells us that there is a unique function $OccurProp\colon Form_P \to \mathcal{P}(P)$ satisfying the necessary requirements. Of course, this method is more precise, but it's significantly less intuitive to use and understand. It's a good exercise to make sure that you can translate a few more

informal recursive definitions in this way, but once you understand how it works you can safely keep the formalism in the back of your mind (at least until we work to develop formal definitions of computation).

Here's a basic example of using induction to prove a result based on a recursive definition.

Proposition 3.1.13. *We have the following:*

1. *If $Q \subseteq P$, then $Form_Q \subseteq Form_P$.*
2. *For any $\varphi \in Form_P$, we have $\varphi \in Form_{OccurProp(\varphi)}$.*

Proof. The first is a straightforward induction on $\varphi \in Form_Q$, and is left as an exercise.

For the second, the proof is by induction on $\varphi \in Form_P$. For the base case, let $\mathsf{A} \in P$ be arbitrary. Since $OccurProp(\mathsf{A}) = \{\mathsf{A}\}$ and $\mathsf{A} \in Form_{\{\mathsf{A}\}}$, we have $\mathsf{A} \in Form_{OccurProp(\mathsf{A})}$.

Let $\varphi \in Form_P$ be such that the statement is true for φ, i.e. such that $\varphi \in Form_{OccurProp(\varphi)}$. Since $OccurProp(\neg\varphi) = OccurProp(\varphi)$, it follows that $\varphi \in Form_{OccurProp(\neg\varphi)}$. Hence, $\neg\varphi \in Form_{OccurProp(\neg\varphi)}$.

Finally, suppose that $\varphi, \psi \in Form_P$, that $\diamond \in \{\wedge, \vee, \rightarrow\}$, and that the statement is true for φ and ψ, i.e. that we have $\varphi \in Form_{OccurProp(\varphi)}$ and $\psi \in Form_{OccurProp(\psi)}$. Since

$$OccurProp(\varphi) \subseteq OccurProp(\diamond\varphi\psi) \quad \text{and} \quad OccurProp(\psi) \subseteq OccurProp(\diamond\varphi\psi)$$

by definition, it follows from (1) that $\varphi, \psi \in Form_{OccurProp(\diamond\varphi\psi)}$. Therefore, $\diamond\varphi\psi \in Form_{OccurProp(\diamond\varphi\psi)}$. □

On to some more important recursive definitions.

Definition 3.1.14. *We define a function $NumConn\colon Form_P \to \mathbb{N}$ recursively as follows:*

- $NumConn(\mathsf{A}) = 0$ *for all* $\mathsf{A} \in P$.
- $NumConn(\neg\varphi) = NumConn(\varphi) + 1$.
- $NumConn(\diamond\varphi\psi) = NumConn(\varphi) + NumConn(\psi) + 1$ *for each* $\diamond \in \{\wedge, \vee, \rightarrow\}$.

Although we have defined propositional formulas as certain finite sequences of symbols, it's more natural to view them as tree structures. The idea is to view a formula like $\wedge\mathsf{AB}$ as a tree having one internal node $\wedge$, and then two leaves as children (with A to the left and B to the right). More complicated formulas lead to more complex trees, but the connectives always serve as internal nodes, and the propositional symbols are the leaves. In computer science, if we view our propositional formulas as code in a programming language, then these trees are called the *syntax trees* of the corresponding formulas. From this perspective, $NumConn$ gives the number of internal nodes of the corresponding tree. Viewing propositional formulas as trees leads to another interesting an fundamental recursive function:

Definition 3.1.15. *We define a function $Depth\colon Form_P \to \mathbb{N}$ recursively as follows:*

- $Depth(\mathsf{A}) = 0$ *for all* $\mathsf{A} \in P$.
- $Depth(\neg\varphi) = Depth(\varphi) + 1$.
- $Depth(\diamond\varphi\psi) = \max\{Depth(\varphi), Depth(\psi)\} + 1$ *for each* $\diamond \in \{\wedge, \vee, \rightarrow\}$.

For example, although $NumConn(\vee \wedge \mathsf{AB} \vee \mathsf{CD}) = 3$, we have $Depth(\vee \wedge \mathsf{AB} \vee \mathsf{CD}) = 2$. Intuitively, the depth of a formula is the the height of the corresponding syntax tree (or 1 off from the height, depending on how you count). If we view these syntax trees as electrical circuits built out of logical gates, then the depth is a good measure of how long it takes the electrical circuit to propagate from the inputs (the propositional symbols at the leaves) to the output (the root of the tree). In other words, an engineer building a circuit to compute as quickly as possible would try to minimize the depth of a circuit, rather than just the number of gates.

Definition 3.1.16. *We define a function* $Subform\colon Form_P \to \mathcal{P}(Form_P)$ *recursively as follows:*

- $Subform(\mathsf{A}) = \{\mathsf{A}\}$ *for all* $\mathsf{A} \in P$.
- $Subform(\neg\varphi) = \{\neg\varphi\} \cup Subform(\varphi)$.
- $Subform(\diamond\varphi\psi) = \{\diamond\varphi\psi\} \cup Subform(\varphi) \cup Subform(\psi)$ *for each* $\diamond \in \{\wedge, \vee, \to\}$.

In other words, $Subform$ takes a formula as input, and produces the set of all formulas that went into building up the formula recursively. For example, we have $Subform(\wedge\neg\mathsf{AB}) = \{\wedge\neg\mathsf{AB}, \neg\mathsf{A}, \mathsf{A}, \mathsf{B}\}$. In terms of the syntax trees, $Subform(\varphi)$ is the set of all subtrees of the syntax tree of φ, obtained by taking each node as a new root. Since our formulas are formally defined as syntactic sequences, a natural question is whether $Subform(\varphi)$ is the set of subsequences of φ that happen to be formulas. This turns out to be true, but is left as an exercise.

We end with a recursive definition of substituting one formula for another.

Definition 3.1.17. *Let* $\theta, \gamma \in Form_P$. *We define a function* $Subst^\theta_\gamma\colon Form_P \to Form_P$ *recursively as follows.*

- $Subst^\theta_\gamma(\mathsf{A}) = \begin{cases} \theta & \text{if } \gamma = \mathsf{A} \\ \mathsf{A} & \text{otherwise.} \end{cases}$
- $Subst^\theta_\gamma(\neg\varphi) = \begin{cases} \theta & \text{if } \gamma = \neg\varphi \\ \neg Subst^\theta_\gamma(\varphi) & \text{otherwise.} \end{cases}$
- $Subst^\theta_\gamma(\diamond\varphi\psi) = \begin{cases} \theta & \text{if } \gamma = \diamond\varphi\psi \\ \diamond Subst^\theta_\gamma(\varphi) Subst^\theta_\gamma(\psi) & \text{otherwise.} \end{cases}$

  for each $\diamond \in \{\wedge, \vee, \to\}$.

For example, we have

$$Subst^{\wedge\mathsf{AB}}_{\neg\mathsf{C}}(\vee\mathsf{C} \to \vee\neg\mathsf{CA}\neg\mathsf{C}) = \vee\mathsf{C} \to \vee \wedge \mathsf{ABA} \wedge \mathsf{AB}.$$

Intuitively, $Subst^\theta_\gamma(\varphi)$ finds all subtrees of the syntax tree of φ that equal the syntax tree of γ, and replaces all of those with the syntax tree of θ.

3.2 Truth Assignments and Semantic Implication

So far, we've treated formulas only as syntactic objects. We briefly alluded to thinking of the syntax tree of a formula as looking like a circuit that computes based on assigning inputs to the propositional symbols, and it is now time to formalize that idea. Recall that, in isolation, a formula like $(\mathsf{A} \wedge \mathsf{B}) \vee \mathsf{C}$ does not have any meaning. However, once we assign true/false values to each of A, B, and C, then the formula obtains a natural truth value. We start with the following definition to codify this assigning of values to the symbols.

Definition 3.2.1. *Given a set* P, *a* truth assignment on P *is a function* $M\colon P \to \{0, 1\}$.

We are using the number 0 and 1 as natural codings of "false" and "true". Once we have a truth assignment M on P, we can define the "truth value" of any formula. Of course, the definition is recursive.

Definition 3.2.2. *Let* P *be a set and let* $M\colon P \to \{0, 1\}$ *be a truth assignment on* P. *We define a function* $v_M\colon Form_P \to \{0, 1\}$ *recursively as follows:*

- $v_M(\mathsf{A}) = M(\mathsf{A})$ *for all* $\mathsf{A} \in P$.
- $v_M(\neg\varphi) = \begin{cases} 1 & \text{if } v_M(\varphi) = 0 \\ 0 & \text{if } v_M(\varphi) = 1. \end{cases}$
- $v_M(\wedge\varphi\psi) = \begin{cases} 0 & \text{if } v_M(\varphi) = 0 \text{ and } v_M(\psi) = 0 \\ 0 & \text{if } v_M(\varphi) = 0 \text{ and } v_M(\psi) = 1 \\ 0 & \text{if } v_M(\varphi) = 1 \text{ and } v_M(\psi) = 0 \\ 1 & \text{if } v_M(\varphi) = 1 \text{ and } v_M(\psi) = 1. \end{cases}$
- $v_M(\vee\varphi\psi) = \begin{cases} 0 & \text{if } v_M(\varphi) = 0 \text{ and } v_M(\psi) = 0 \\ 1 & \text{if } v_M(\varphi) = 0 \text{ and } v_M(\psi) = 1 \\ 1 & \text{if } v_M(\varphi) = 1 \text{ and } v_M(\psi) = 0 \\ 1 & \text{if } v_M(\varphi) = 1 \text{ and } v_M(\psi) = 1. \end{cases}$
- $v_M(\rightarrow\varphi\psi) = \begin{cases} 1 & \text{if } v_M(\varphi) = 0 \text{ and } v_M(\psi) = 0 \\ 1 & \text{if } v_M(\varphi) = 0 \text{ and } v_M(\psi) = 1 \\ 0 & \text{if } v_M(\varphi) = 1 \text{ and } v_M(\psi) = 0 \\ 1 & \text{if } v_M(\varphi) = 1 \text{ and } v_M(\psi) = 1. \end{cases}$

Before moving on, we should note a couple of simple facts about what happens when we either shrink or enlarge the set P. Intuitively, if $\varphi \in Form_Q$ and $Q \subseteq P$, then we can extend the truth assignment from Q to P arbitrarily without affecting the value of $v_M(\varphi)$. In fact, it seems clear that the value $v_M(\varphi)$ depends only on the values $M(\mathsf{A})$ for those A that actually occur in φ. The next result formalizes these ideas.

Proposition 3.2.3. *Let P be a set.*

1. *Suppose that $Q \subseteq P$ and that $M\colon P \to \{0,1\}$ is a truth assignment on P. For all $\varphi \in Form_Q$, we have $v_M(\varphi) = v_{M\restriction Q}(\varphi)$.*
2. *Suppose $\varphi \in Form_P$. Whenever M and N are truth assignments on P such that $M(\mathsf{A}) = N(\mathsf{A})$ for all $\mathsf{A} \in OccurProp(\varphi)$, we have $v_M(\varphi) = v_N(\varphi)$.*

Proof. The first part follows by a straightforward induction on $\varphi \in Form_Q$, and is left as an exercise. For the second, let $Q = OccurProp(\varphi)$. We then have that $\varphi \in Form_Q$ by Proposition 3.1.13, and so

$$\begin{aligned} v_M(\varphi) &= v_{M\restriction Q}(\varphi) && \text{(by part (1))} \\ &= v_{N\restriction Q}(\varphi) && \text{(since } M \restriction Q = N \restriction Q\text{)} \\ &= v_N(\varphi) && \text{(by part (1)).} \end{aligned}$$

This completes the proof. □

With a method of assigning true/false values to formulas in hand (once we've assigned them to P), we are now in position to use our semantic definitions to given a precise meaning to "The set of formulas Γ implies the formula φ".

Definition 3.2.4. *Let P be a set, let $\Gamma \subseteq Form_P$,and let $\varphi \in Form_P$. We write $\Gamma \vDash_P \varphi$ to mean that whenever M is a truth assignment on P with the property that $v_M(\gamma) = 1$ for all $\gamma \in \Gamma$, we have $v_M(\varphi) = 1$. We pronounce $\Gamma \vDash_P \varphi$ as* Γ semantically implies φ.

For example, if $P = \{\mathsf{A}, \mathsf{B}, \mathsf{C}\}$, then we claim that $\{\mathsf{A} \vee \mathsf{B}, \neg(\mathsf{A} \wedge (\neg\mathsf{C}))\} \models_P \mathsf{B} \vee \mathsf{C}$. To see this, let $M\colon P \to \{0,1\}$ be a truth assignment such that $v_M(\mathsf{A} \vee \mathsf{B}) = 1$ and $v_M(\neg(\mathsf{A} \wedge (\neg\mathsf{C}))) = 1$. We then have $v_M(\mathsf{A} \wedge (\neg\mathsf{C})) = 0$, so either $v_M(\mathsf{A}) = 0$ or $v_M(\neg\mathsf{C}) = 0$. We handle the two cases:

- *Case 1:* Suppose that $v_M(\mathsf{A}) = 0$. Since $v_M(\mathsf{A} \vee \mathsf{B}) = 1$, it follows that $v_M(\mathsf{B}) = 1$, and hence $v_M(\mathsf{B} \vee \mathsf{C}) = 1$.

- *Case 2:* Suppose that $v_M(\neg\mathsf{C}) = 0$. We then have $v_M(\mathsf{C}) = 1$, and hence $v_M(\mathsf{B} \vee \mathsf{C}) = 1$.

Therefore, we have $v_M(\mathsf{B} \vee \mathsf{C}) = 1$ in either case. It follows that $\{\mathsf{A} \vee \mathsf{B}, \neg(\mathsf{A} \wedge (\neg\mathsf{C}))\} \models \mathsf{B} \vee \mathsf{C}$.

For another example, we claim that for any set P and any $\varphi, \psi \in Form_P$, we have $\{\varphi \to \psi, \varphi\} \models_P \psi$. Again, let $M\colon P \to \{0,1\}$ be an arbitrary truth assignment such that both $v_M(\varphi \to \psi) = 1$ and $v_M(\varphi) = 1$. If $v_M(\psi) = 0$, it would follows that $v_M(\varphi \to \psi) = 0$, a contradiction. Therefore, $v_M(\psi) = 1$.

As above, it is intuitively clear that in order to understand whether $\Gamma \models_P \varphi$, we need only think about the values of the truth assignments $M\colon P \to \{0,1\}$ on the symbols that actually appear in $\Gamma \cup \{\varphi\}$.

Proposition 3.2.5. *Suppose that $Q \subseteq P$, that $\Gamma \subseteq Form_Q$, and that $\varphi \in Form_Q$. We then have that $\Gamma \models_P \varphi$ if and only if $\Gamma \models_Q \varphi$.*

Proof. First notice that $\Gamma \subseteq Form_P$ and $\varphi \in Form_P$ by Proposition 3.1.13.

Suppose first that $\Gamma \models_Q \varphi$. Let $M\colon P \to \{0,1\}$ be a truth assignment such that $v_M(\gamma) = 1$ for all $\gamma \in \Gamma$. By Proposition 3.2.3, we then have that $v_{M\restriction Q}(\gamma) = 1$ for all $\gamma \in \Gamma$. Since $\Gamma \models_Q \varphi$, it follows that $v_{M\restriction Q}(\varphi) = 1$. Using Proposition 3.2.3 again, we conclude that $v_M(\varphi) = 1$. Therefore, $\Gamma \models_P \varphi$.

Suppose conversely that $\Gamma \models_P \varphi$. Let $M\colon Q \to \{0,1\}$ be a truth assignment such that $v_M(\gamma) = 1$ for all $\gamma \in \Gamma$. Define a truth assignment $N\colon P \to \{0,1\}$ by letting $N(\mathsf{A}) = M(\mathsf{A})$ for all $\mathsf{A} \in Q$ and letting $N(\mathsf{A}) = 0$ for all $\mathsf{A} \in P \backslash Q$. Since $N \restriction Q = M$, Proposition 3.2.3 implies that $v_N(\gamma) = v_M(\gamma) = 1$ for all $\gamma \in \Gamma$. Since $\Gamma \models_P \varphi$, it follows that $v_N(\varphi) = 1$. Using Proposition 3.2.3 again, we conclude that $v_M(\varphi) = 1$. Therefore, $\Gamma \models_Q \varphi$. □

Since the ambient set P does not matter, we will almost always omit the P in $\models_P$. We also adopt the following conventions in order to simplify notation.

Notation 3.2.6. *Let P be a set.*

1. *If $\Gamma = \emptyset$, we write $\models \varphi$ instead of $\emptyset \models \varphi$.*

2. *If $\Gamma = \{\gamma\}$, we write $\gamma \models \varphi$ instead of $\{\gamma\} \models \varphi$.*

Given a finite set $\Gamma \subseteq Form_P$ and a formula $\varphi \in Form_P$, there is a basic procedure that we can always follow in order to determine whether $\Gamma \models \varphi$. By Proposition 3.2.5, instead of examining all truth assignments on P, we need only consider truth assignments on the finite set Q of propositional symbols that actually occur in $\Gamma \cup \{\varphi\}$. Since Q is a finite set, there are only finitely many such truth assignments. Thus, one way of determining whether $\Gamma \models \varphi$ is simply to check them all. We can systematically arrange the truth assignments in a table (see below), where we ensure that we put the the elements of Q in the first columns, and put all elements of $\Gamma \cup \{\varphi\}$ in later columns. We also ensure that if ψ is in a column, then all subformulas of ψ appear in an earlier column, which allows us to fill in the table one column at a time. This simple-minded exhaustive technique is called the method of *truth tables*.

For example, suppose that we want to show that $\{(\mathsf{A} \vee \mathsf{B}) \wedge \mathsf{C}, \mathsf{A} \to (\neg\mathsf{C})\} \models (\neg\mathsf{C}) \to \mathsf{B}$. We build the following table:

A	B	C	A ∨ B	(A ∨ B) ∧ C	¬C	A → (¬C)	(¬C) → B
0	0	0	0	0	1	1	0
0	0	1	0	0	0	1	1
0	1	0	1	0	1	1	1
0	1	1	1	1	0	1	1
1	0	0	1	0	1	1	0
1	0	1	1	1	0	0	1
1	1	0	1	0	1	1	1
1	1	1	1	1	0	0	1

Notice that in every row where both the $(\mathsf{A} \vee \mathsf{B}) \wedge \mathsf{C}$ column and the $\mathsf{A} \to (\neg\mathsf{C})$ column have a 1, namely just the row beginning with 011, we have that the entry under the $(\neg\mathsf{C}) \to \mathsf{B}$ column is a 1. Therefore, $\{(\mathsf{A} \vee \mathsf{B}) \wedge \mathsf{C}, \mathsf{A} \to (\neg\mathsf{C})\} \vDash (\neg\mathsf{C}) \to \mathsf{B}$.

Definition 3.2.7.

1. *Let $\varphi \in Form_P$. We say that φ is a* tautology *if $\vDash \varphi$.*

2. *If $\varphi \vDash \psi$ and $\psi \vDash \varphi$, we say that φ and ψ are* semantically equivalent.

The formula $(\mathsf{A} \wedge \mathsf{B}) \to (\mathsf{A} \vee \mathsf{B})$ is a tautology, as is $\mathsf{A} \vee (\neg\mathsf{A})$. In fact, for any formula $\varphi \in Form_P$, the formula $\varphi \vee (\neg\varphi)$ is a tautology.

In terms of truth tables, to check that φ is a tautology, we simply check that every entry in the column of φ is a 1. To check that φ and ψ are semantically equivalent, the next result states that we can examine whether the entries in the column of φ equal the corresponding entries in the column of ψ.

Proposition 3.2.8. *Let $\varphi, \psi \in Form_P$. The following are equivalent:*

1. *φ and ψ are semantically equivalent.*

2. *For all truth assignments $M\colon P \to \{0,1\}$, we have $v_M(\varphi) = 1$ if and only if $v_M(\psi) = 1$.*

3. *For all truth assignments $M\colon P \to \{0,1\}$, we have $v_M(\varphi) = 0$ if and only if $v_M(\psi) = 0$.*

4. *For all truth assignments $M\colon P \to \{0,1\}$, we have $v_M(\varphi) = v_M(\psi)$.*

Proof. By definition, φ and ψ are semantically equivalent if and only if both $\varphi \vDash \psi$ and $\psi \vDash \varphi$. In other words, φ and ψ are semantically equivalent if and only if whenever $M\colon P \to \{0,1\}$ is a truth assignment, then either both $v_M(\varphi) = 1$ and $v_M(\psi) = 1$ are true, or both are false. Since the only possible values of v_M are 0 and 1, this is equivalent to saying that $v_M(\varphi) = v_M(\psi)$ for all truth assignments $M\colon P \to \{0,1\}$. □

For example, we claim that $\neg(\mathsf{A} \wedge \mathsf{B})$ is semantically equivalent to $\neg\mathsf{A} \vee \neg\mathsf{B}$. Here is the corresponding truth table that includes both formulas $\neg(\mathsf{A} \wedge \mathsf{B})$ and $\neg\mathsf{A} \vee \neg\mathsf{B}$:

A	B	A ∧ B	¬(A ∧ B)	¬A	¬B	¬A ∨ ¬B
0	0	0	1	1	1	1
0	1	0	1	1	0	1
1	0	0	1	0	1	1
1	1	1	0	0	0	0

Notice that the rows in which the $\neg(\mathsf{A} \wedge \mathsf{B})$ column has a 1 are exactly the same as the rows in which the $\neg\mathsf{A} \vee \neg\mathsf{B}$ column has a 1. Therefore, $\neg(\mathsf{A} \wedge \mathsf{B})$ is semantically equivalent to $\neg\mathsf{A} \vee \neg\mathsf{B}$.

For any $\varphi \in Form_P$, it's easy to see that the formulas φ and $\neg\neg\varphi$ are semantically equivalent. Here is a slightly more interesting example: The formulas $\varphi \to \psi$ and $\neg\varphi \vee \psi$ are semantically equivalent for any $\varphi, \psi \in Form_P$. To see this, notice that given an arbitrary truth assignment $M\colon P \to \{0,1\}$, we have

$$\begin{aligned} v_M(\varphi \to \psi) = 1 &\Leftrightarrow v_M(\varphi) = 0 \text{ or } v_M(\psi) = 1 \\ &\Leftrightarrow v_M(\neg\varphi) = 1 \text{ or } v_M(\psi) = 1 \\ &\Leftrightarrow v_M(\neg\varphi \vee \psi) = 1. \end{aligned}$$

Thus, $v_M(\varphi \to \psi) = v_M(\neg\varphi \vee \psi)$ for all truth assignments $M\colon P \to \{0,1\}$. Alternatively, we could build a truth table like above, except starting with φ and ψ in the first columns rather than specific propositional symbols.

Since $\varphi \to \psi$ and $\neg\varphi \vee \psi$ are semantically equivalent for any $\varphi, \psi \in Form_P$, it seems redundant to include the symbol $\to$. Intuitively, we can replace every occurrence of $\to$ using this rule without affecting the "meaning" of the formula. We now prove a formal version of this statement.

Proposition 3.2.9. *Let $Form_P^-$ be the subset of $Form_P$ obtained by omitting $h_\to$ from the generating system, i.e. $Form_P^- = G(Sym_P^*, P, \{h_\neg, h_\wedge, h_\vee\})$. For all $\varphi \in Form_P$, there exists $\psi \in Form_P^-$ such that φ and ψ are semantically equivalent.*

Proof. Define a function $h\colon Form_P \to Form_P^-$ recursively as follows:

- $h(\mathsf{A}) = \mathsf{A}$ for all $\mathsf{A} \in P$.
- $h(\neg\varphi) = \neg h(\varphi)$ for all $\varphi \in Form_P$.
- $h(\wedge\varphi\psi) = \wedge h(\varphi)h(\psi)$ for all $\varphi, \psi \in Form_P$.
- $h(\vee\varphi\psi) = \vee h(\varphi)h(\psi)$ for all $\varphi, \psi \in Form_P$.
- $h(\to \varphi\psi) = \vee\neg h(\varphi)h(\psi)$ for all $\varphi, \psi \in Form_P$.

We prove that φ and $h(\varphi)$ are semantically equivalent for all $\varphi \in Form_P$ by induction. In other words, we let

$$X = \{\varphi \in Form_P : \varphi \text{ and } h(\varphi) \text{ are semantically equivalent}\},$$

and show that $X = Form_P$ by proving that X is an inductive set. For the base case, we have $h(\mathsf{A}) = \mathsf{A}$ for all $\mathsf{A} \in P$ by definition, so trivially we have that A and $h(\mathsf{A})$ are semantically equivalent for all $\mathsf{A} \in P$. We now handle the four inductive steps.

- Let $\varphi \in Form_P$ be arbitrary such that φ and $h(\varphi)$ are semantically equivalent. Let $M\colon P \to \{0,1\}$ be an arbitrary truth assignment. We have

  $$\begin{aligned} v_M(h(\neg\varphi)) = 1 &\Leftrightarrow v_M(\neg h(\varphi)) = 1 && \text{(by definition of } h) \\ &\Leftrightarrow v_M(h(\varphi)) = 0 \\ &\Leftrightarrow v_M(\varphi) = 0 && \text{(by Proposition 3.2.8)} \\ &\Leftrightarrow v_M(\neg\varphi) = 1. \end{aligned}$$

  Using Proposition 3.2.8, we conclude that $\neg\varphi$ and $h(\neg\varphi)$ are semantically equivalent.

- Let $\varphi, \psi \in Form_P$ be arbitrary such that φ and $h(\varphi)$ are semantically equivalent, and such that ψ and $h(\psi)$ are semantically equivalent. Let $M\colon P \to \{0,1\}$ be an arbitrary truth assignment. We have

  $$\begin{aligned} v_M(h(\wedge\varphi\phi)) = 1 &\Leftrightarrow v_M(\wedge h(\varphi)h(\psi)) = 1 && \text{(by definition of } h) \\ &\Leftrightarrow v_M(h(\varphi)) = 1 \text{ and } v_M(h(\psi)) = 1 \\ &\Leftrightarrow v_M(\varphi) = 1 \text{ and } v_M(\psi) = 1 && \text{(by Proposition 3.2.8)} \\ &\Leftrightarrow v_M(\wedge\varphi\psi) = 1. \end{aligned}$$

Using Proposition 3.2.8, we conclude that $\wedge\varphi\phi$ and $h(\wedge\varphi\psi)$ are semantically equivalent.

- Let $\varphi, \psi \in Form_P$ be arbitrary such that φ and $h(\varphi)$ are semantically equivalent, and such that ψ and $h(\psi)$ are semantically equivalent. Let $M\colon P \to \{0,1\}$ be an arbitrary truth assignment. We have

$$\begin{aligned} v_M(h(\vee\varphi\phi)) = 1 &\Leftrightarrow v_M(\vee h(\varphi)h(\psi)) = 1 && \text{(by definition of } h\text{)} \\ &\Leftrightarrow \text{Either } v_M(h(\varphi)) = 1 \text{ or } v_M(h(\psi)) = 1 \\ &\Leftrightarrow \text{Either } v_M(\varphi) = 1 \text{ or } v_M(\psi) = 1 && \text{(by Proposition 3.2.8)} \\ &\Leftrightarrow v_M(\vee\varphi\psi) = 1. \end{aligned}$$

Using Proposition 3.2.8, we conclude that $\vee\varphi\phi$ and $h(\vee\varphi\psi)$ are semantically equivalent.

- Let $\varphi, \psi \in Form_P$ be arbitrary such that φ and $h(\varphi)$ are semantically equivalent, and such that ψ and $h(\psi)$ are semantically equivalent. Let $M\colon P \to \{0,1\}$ be an arbitrary truth assignment. We have

$$\begin{aligned} v_M(h(\to\varphi\phi)) = 1 &\Leftrightarrow v_M(\vee\neg h(\varphi)h(\psi)) = 1 && \text{(by definition of } h\text{)} \\ &\Leftrightarrow \text{Either } v_M(\neg h(\varphi)) = 1 \text{ or } v_M(h(\psi)) = 1 \\ &\Leftrightarrow \text{Either } v_M(h(\varphi)) = 0 \text{ or } v_M(h(\psi)) = 1 \\ &\Leftrightarrow \text{Either } v_M(\varphi) = 0 \text{ or } v_M(\psi) = 1 && \text{(by Proposition 3.2.8)} \\ &\Leftrightarrow v_M(\to\varphi\psi) = 1. \end{aligned}$$

Using Proposition 3.2.8, we conclude that $\to\varphi\phi$ and $h(\to\varphi\psi)$ are semantically equivalent.

By induction, it follows that φ and $h(\varphi)$ are semantically equivalent for all $\varphi \in Form_P$. □

In fact, it is possible to improve this result in several orthogonal ways. For instance, since $\varphi \vee \psi$ and $\neg((\neg\varphi) \wedge (\neg\psi))$ are semantically equivalent for any $\varphi, \psi \in Form_P$, a similar argument to Proposition 3.2.9 shows that we can also do away with the function $h_\vee$. In other words, every element of $Form_P$ is semantically equivalent to a formula in $G(Sym_P^*, P, \{h_\neg, h_\wedge\})$. We can instead choose to eliminate the $\wedge$ connective. That is, since $\varphi \wedge \psi$ and $\neg((\neg\varphi) \vee (\neg\psi))$ are semantically equivalent for any $\varphi, \psi \in Form_P$, it follows that element of $Form_P$ is semantically equivalent to a formula in $G(Sym_P^*, P, \{h_\neg, h_\vee\})$.

We can also follows a middle way by keeping both the $\wedge$ and $\vee$ connectives, but only allow a very limited version of negations. In particular, we can restrict negations to only apply to the propositional symbols directly. Here is the formal definition.

Definition 3.2.10. *A* literal *is a element of* $P \cup \{\neg\mathsf{A} : \mathsf{A} \in P\}$. *We denote the set of literals by* $Literal_P$.

Now we build up our restricted formulas by starting with the literals, and then generating using only $h_\wedge$ and $h_\vee$. Following a recursive construction similar to the proof of Proposition 3.2.9, one can show the following.

Proposition 3.2.11. *For all* $\varphi \in Form_P$, *there exists* $\psi \in G(Sym_P^*, Literal_P, \{h_\vee, h_\wedge\})$ *such that* φ *and* ψ *are semantically equivalent.*

Proof. Exercise, although we'll see a nonrecursive way to prove this fact in the next section. □

Finally, we end this section with a semantic way to say that a set of formulas is not contradictory.

Definition 3.2.12. *Let* P *be a set and let* $\Gamma \subseteq Form_P$. *We say that* Γ *is* satisfiable *if there exists a truth assignment* $M\colon P \to \{0,1\}$ *such that* $v_M(\gamma) = 1$ *for all* $\gamma \in \Gamma$. *Otherwise, we say that* Γ *is* unsatisfiable.

For example, the set

$$\{\mathsf{A} \vee (\mathsf{B} \wedge \mathsf{C}), \mathsf{A} \to (\neg \mathsf{C}), \mathsf{B} \to \mathsf{C}\}$$

is satisfiable, as witnessed by the truth assignment $M\colon P \to \{0,1\}$ defined by $M(\mathsf{A}) = 1$, $M(\mathsf{B}) = 0$, and $M(\mathsf{C}) = 0$. In contrast, the set

$$\{\mathsf{A} \wedge (\mathsf{B} \vee \mathsf{C}), \mathsf{A} \to (\neg \mathsf{C}), \mathsf{B} \to \mathsf{C}\}$$

is unsatisfiable, which can be verified by a simple exhaustive check. In general, determining if a finite set of formulas is satisfiable is very difficult. In fact, the computational problem that consists of taking a finite set of formulas, and determining whether it is satisfiable, is one of the most important problems in computer science. We know of no efficient method that works in general. Any fast algorithm that solves the satisfiability problem can be repurposed to solve an enormous number of seemingly disparate problems throughout computer science (all problems in the complexity class NP). Unfortunately, we can't delve into the theory of NP-completeness here.

3.3 Boolean Functions and Connectives

After seeing that some of our connectives are unnecessary (i.e. can be removed without affecting the expressive power of our formulas), it is natural to wonder if our choice of connectives is the "right" one. For example, why didn't we introduce a new connective $\leftrightarrow$, allow ourselves to build the formula $\varphi \leftrightarrow \psi$ (or $\leftrightarrow \varphi\psi$ in Polish notation) whenever $\varphi, \psi \in Form_P$, and then extend our recursive definition of v_M so that

$$v_M(\leftrightarrow \varphi\psi) = \begin{cases} 1 & \text{if } v_M(\varphi) = 0 \text{ and } v_M(\psi) = 0 \\ 0 & \text{if } v_M(\varphi) = 0 \text{ and } v_M(\psi) = 1 \\ 0 & \text{if } v_M(\varphi) = 1 \text{ and } v_M(\psi) = 0 \\ 1 & \text{if } v_M(\varphi) = 1 \text{ and } v_M(\psi) = 1. \end{cases}$$

Of course, there is no real need to introduce this connective because for any $\varphi, \psi \in Form_P$ we would have that $\varphi \leftrightarrow \psi$ is semantically equivalent to $(\varphi \to \psi) \wedge (\psi \to \varphi)$. Using a recursive construction analogous to the proof of Proposition 3.2.9, it follows that every formula with this expanded connective is semantically equivalent to one without it.

Perhaps we could be more exotic and introduce a new connective $\square$ that allows us to build the formula $\square\varphi\psi\theta$ (here's an instance when Polish notation becomes important) whenever $\varphi, \psi, \theta \in Form_P$, and extends our definition of v_M so that

$$v_M(\square\varphi\psi\theta) = \begin{cases} 1 & \text{if at least two of } v_M(\varphi) = 1, v_M(\psi) = 1, \text{ and } v_M(\theta) = 1 \\ 0 & \text{otherwise.} \end{cases}$$

In other words, our new connective $\square$ is the "majority" connective, i.e. it evaluates the individual truth values of the three formulas, and outputs the one that occurs most. It's not hard (and a good exercise) to show that for any $\varphi, \psi, \theta \in Form_P$, there exists $\alpha \in Form_P$ such that $\square\varphi\psi\theta$ is semantically equivalent to α. From here, we can again show that every formula with this expanded connective is semantically equivalent to one without it.

We want a general theorem which says that no matter how exotic a connective one invents, it's always possible to find an element of $Form_P$ which is semantically equivalent, and thus our choice of connectives is sufficient to express everything we'd ever want. Rather than deal with arbitrary connectives, the real issue here is whether we can express any possible function taking k true/false values to true/false values.

Definition 3.3.1. *Let $k \in \mathbb{N}^+$. A function $f\colon \{0,1\}^k \to \{0,1\}$ is called a* boolean function of arity k.

Definition 3.3.2. *Suppose that* $P = \{\mathsf{A}_0, \mathsf{A}_1, \ldots, \mathsf{A}_{k-1}\}$. *Given* $\varphi \in Form_P$, *we define a boolean function* $B_\varphi \colon \{0,1\}^k \to \{0,1\}$ *as follows. Given* $\sigma \in \{0,1\}^k$, *define a truth assignment* $M \colon P \to \{0,1\}$ *by letting* $M(\mathsf{A}_i) = \sigma(i)$ *for all* i, *and set* $B_\varphi(\sigma) = v_M(\varphi)$.

Notice that when $P = \{\mathsf{A}_0, \mathsf{A}_1, \ldots, \mathsf{A}_{k-1}\}$, then given arbitrary $\varphi, \psi \in Form_P$, we have that φ and ψ are semantically equivalent if and only if $B_\varphi = B_\psi$, because as we vary $\sigma \in \{0,1\}^k$, we are covering all possible truth assignments. We now show that we can express all possible boolean functions using the connectives that we have.

Theorem 3.3.3. *Let* $k \in \mathbb{N}^+$ *be arbitrary, and let* $P = \{\mathsf{A}_0, \mathsf{A}_1, \ldots, \mathsf{A}_{k-1}\}$. *For any boolean function* $f \colon \{0,1\}^k \to \{0,1\}$ *of arity* k, *there exists* $\varphi \in Form_P$ *such that* $f = B_\varphi$.

In fact, we'll prove a stronger theorem below which says that we may assume that our formula φ is in a particularly simple form. Before diving into the general argument, let's look at an example. Suppose that $f \colon \{0,1\}^3 \to \{0,1\}$ is given by the following table:

0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

Suppose we wanted to come up with a formula φ such that $f = B_\varphi$. One option is to use a lot of thought to come up with an elegant solution. Another is simply to think as follows. Since $f(000) = 1$, perhaps we should put

$$\neg\mathsf{A}_0 \wedge \neg\mathsf{A}_1 \wedge \neg\mathsf{A}_2$$

into the formula somewhere. This subformula will "light up" on the input 000, but not on any other inputs. Similarly, since $f(010) = 1$, we could imagine putting

$$\neg\mathsf{A}_0 \wedge \mathsf{A}_1 \wedge \neg\mathsf{A}_2$$

into the formula somewhere to activate on the input 010. If we do the same to the other lines which have value 1, we can put all of these pieces together in a manner which makes them all play nice by connecting them with $\vee$. Thus, our formula is

$$(\neg\mathsf{A}_0 \wedge \neg\mathsf{A}_1 \wedge \neg\mathsf{A}_2) \vee (\neg\mathsf{A}_0 \wedge \mathsf{A}_1 \wedge \neg\mathsf{A}_2) \vee (\mathsf{A}_0 \wedge \mathsf{A}_1 \wedge \neg\mathsf{A}_2) \vee (\mathsf{A}_0 \wedge \mathsf{A}_1 \wedge \mathsf{A}_2).$$

Since we connecting the various subformulas with the $\vee$ connective, the entire formula will output 1 exactly when at least one of our special subformulas outputs a 1.

Notice the special form of the formula that we have produced in our previous example. We only applied the negation symbols to propositional symbols, i.e. the only use of negations was to form literals. From the literals, we applied the $\wedge$ connective repeatedly to form the subformulas. And then from these formulas we applied the $\vee$ connective repeatedly to form our formula. The formulas that can be obtained in this way are said to be in *disjunctive normal form*. Here is the formal definition.

Definition 3.3.4. *Let* P *be a set. We let* $Conj_P = G(Sym_P^*, Literal_P, \{h_\wedge\})$ *be the formulas obtained by starting with the literals and generating using only* $h_\wedge$, *and call* $Conj_P$ *the set of* conjunctive formulas. *From here, we define* $DNF_P = G(Sym_P^*, Conj_P, \{h_\vee\})$ *to be the formulas obtained by starting with the conjunctive formulas, and generating using only* $h_\vee$. *The elements of* DNF_P *are said to be in* disjunctive normal form.

We now prove the following theorem, which trivially implies Theorem 3.3.3.

Theorem 3.3.5. *Let $k \in \mathbb{N}^+$ be arbitrary, and let $P = \{\mathsf{A}_0, \mathsf{A}_1, \ldots, \mathsf{A}_{k-1}\}$. For any boolean function $f\colon \{0,1\}^k \to \{0,1\}$ of arity k, there exists $\varphi \in DNF_P$ such that $f = B_\varphi$.*

Proof. Let $T = \{\sigma \in \{0,1\}^k : f(\sigma) = 1\}$. If $T = \emptyset$, we may let φ be $\mathsf{A}_0 \wedge (\neg \mathsf{A}_0)$, which is trivially an element of DNP_P. Suppose then that $T \neq \emptyset$. For each $\sigma \in T$, let

$$\psi_\sigma = \bigwedge_{i=0}^{k-1} \theta_i,$$

where

$$\theta_i = \begin{cases} \mathsf{A}_i & \text{if } \sigma(i) = 1 \\ \neg \mathsf{A}_i & \text{if } \sigma(i) = 0. \end{cases}$$

For each $\sigma \in T$, notice that $\psi_\sigma \in Conj_P$ because $\theta_i \in Literal_P$ for all i. Finally, let

$$\varphi = \bigvee_{\sigma \in T} \psi_\sigma$$

and notice that $\varphi \in DNF_P$. We then have that $f = B_\varphi$. □

3.4 Syntactic Implication

We now work to define a different notion of implication, which is based on syntactic manipulations instead of a detour through truth assignments and other semantic notions. Our goal is to set up a "proof system" that states the basic implications that we are allowed to write down, and then gives rules about how to transform certain implications into other implications. Although we will fix a choice of basic implications and transformation rules, there are many other choices that one can make. Some approaches pride themselves on being minimalistic by using very few basic implications and rules, often at the expense of making the system extremely unnatural to work with (but easier to prove things about!). We'll take a different approach and set down our rules based on the types of steps in a proof that are used naturally throughout mathematics.

Since we will want our rules to be simple and mechanistic, we will ensure that everything in sight is finite and easily coded by a computer. The objects that we will manipulate will be pairs, where the first component is a finite sequence of formulas, and the second is a formula. Given a finite sequence $S \in Form_P^*$ and a formula $\varphi \in Form_P$, we will write $S \vdash \varphi$ to intuitively mean that there is a formal syntactic proof of φ from the assumptions that appear in the sequence S. We begin with the most basic proofs.

Trivial Implications: We can assert $S \vdash \varphi$ if φ appears as an element in the sequence S, i.e. if there exists an $i < |S|$ such that $S(i) = \gamma$. We denote these uses of this by writing $(Assume_P)$, since our conclusion appears in our assumptions.

With these in hand, we describe ways to generate new formal proof from ones that we already have established. To ease notation, we will write S, γ to mean that we add γ as a new element onto the end of the sequence S. In other words, S, γ means that we concatenate S with the one element sequence γ. In each case, we interpret these rules as follows: If we have already established the formal proof(s) appearing above the horizontal line, then we are allowed to conclude the formal proof appearing below the horizontal line.

Rules for $\wedge$: We have two rules for $\wedge$-elimination and one for $\wedge$-introduction:

$$\frac{S \vdash \varphi \wedge \psi}{S \vdash \varphi} \ (\wedge EL) \qquad \frac{S \vdash \varphi \wedge \psi}{S \vdash \psi} \ (\wedge ER) \qquad \frac{S \vdash \varphi \quad S \vdash \psi}{S \vdash \varphi \wedge \psi} \ (\wedge I)$$

Rules for $\vee$: We have two rules for introducing $\vee$:

$$\frac{S \vdash \varphi}{S \vdash \varphi \vee \psi} \ (\vee IL) \qquad \frac{S \vdash \psi}{S \vdash \varphi \vee \psi} \ (\vee IR)$$

Rules for $\rightarrow$: We have two rules here, one for elimination and for introduction:

$$\frac{S \vdash \varphi \rightarrow \psi}{S, \varphi \vdash \psi} \ (\rightarrow E) \qquad \frac{S, \varphi \vdash \psi}{S \vdash \varphi \rightarrow \psi} \ (\rightarrow I)$$

Rules for proofs by cases: We have two ways to give an argument based on cases:

$$\frac{S, \varphi \vdash \theta \quad S, \psi \vdash \theta}{S, \varphi \vee \psi \vdash \theta} \ (\vee PC) \qquad \frac{S, \psi \vdash \varphi \quad S, \neg\psi \vdash \varphi}{S \vdash \varphi} \ (\neg PC)$$

Rule for proof by contradiction:

$$\frac{S, \neg\varphi \vdash \psi \quad S, \neg\varphi \vdash \neg\psi}{S \vdash \varphi} \ (Contr)$$

Assumption transformation rules:

$$\frac{S \vdash \varphi}{S, \gamma \vdash \varphi} \ (Expand) \qquad \frac{S, \gamma, \gamma \vdash \varphi}{S, \gamma \vdash \varphi} \ (Delete) \qquad \frac{S_1, \gamma_1, \gamma_2, S_2 \vdash \varphi}{S_1, \gamma_2, \gamma_1, S_2 \vdash \varphi} \ (Reorder)$$

As alluded to above, the idea is to start with the trivial formal proofs where the conclusion is included in the assumptions, and the use the given rules to generate new formal proofs. For a very simple of how these rules can be iterated to form more complex implications, consider the following layering of two rules:

$$\begin{array}{lll} \mathsf{A} \wedge \mathsf{B} \vdash \mathsf{A} \wedge \mathsf{B} & (Assume_P) & (1) \\ \mathsf{A} \wedge \mathsf{B} \vdash \mathsf{A} & (\wedge EL \text{ on } 1) & (2) \\ \mathsf{A} \wedge \mathsf{B} \vdash \mathsf{A} \vee \mathsf{B} & (\vee I \text{ on } 2) & (3) \end{array}$$

Therefore, we conclude that $\mathsf{A} \wedge \mathsf{B} \vdash \mathsf{A} \vee \mathsf{B}$. From this example, it's clear that we are generating new implications from others, so we can really view this situation as a generating system. Each line of an argument like the one above is a pair, consisting of a sequence from $Form_P$ and an element of $Form_P$, leading us to the following definition.

Definition 3.4.1. *Let* $Line_P = Form_P^* \times Form_P$.

When viewing the formal proofs as the elements that arise from a generating system, we need to define the set of elements that we start generating from.

Definition 3.4.2. *Let* $Assume_P = \{(S, \varphi) \in Line_P : \text{There exists } i < |S| \text{ such that } S(i) = \varphi\}$.

We next need to interpret the various rules as arising from functions in our generating system. In this case, it's most natural to define a generating system that is not simple, although it's possible to hack together a simple one that works as well. For an example of one of the functions in our (non-simple) generating system, we define $h_{\wedge EL} \colon Line_P \rightarrow \mathcal{P}(Line_P)$ as follows:

$$h_{\wedge EL}(S, \alpha) = \begin{cases} \{(S, \varphi)\} & \text{if there exists } \varphi, \psi \in Form_P \text{ with } \alpha = \varphi \wedge \psi \\ \emptyset & \text{otherwise.} \end{cases}$$

Notice that the above definition makes sense because the system that generated formulas was free, so if there exists $\varphi, \psi \in Form_P$ with $\alpha = \varphi \wedge \psi$, then there is a unique such choice. The definition of $h_{\wedge EL}$ is similar.

For the $\wedge I$ rule, we define a function $h_{\wedge I}\colon (Line_P)^2 \to \mathcal{P}(Line_P)$ as follows:

$$h_{\wedge I}((S_1,\varphi_1),(S_2,\varphi_2)) = \begin{cases} \{(S_1, \varphi_1 \wedge \varphi_2)\} & \text{if } S_1 = S_2 \\ \emptyset & \text{otherwise.} \end{cases}$$

For the $\vee IL$ rule, we define a function $h_{\vee IL}\colon Line_P \to \mathcal{P}(Line_P)$ as follows:

$$h_{\vee IL}(S,\varphi) = \{(S, \varphi \vee \psi) : \psi \in Form_P\}.$$

We define $h_{\vee IR}$ similarly. It is more complicated, but reasonably straightforward, to write down functions for the other rules. Letting $\mathcal{H}$ be the collection of all of such functions, we arrive at the following definition.

Definition 3.4.3. *Let $S \in Form_P^*$ and let $\varphi \in Form_P$. We write $S \vdash \varphi$ to mean that*

$$(S,\varphi) \in G(Line_P, Assume_P, \mathcal{H}).$$

Let's go back and examine our argument showing that $\mathsf{A} \wedge \mathsf{B} \vdash \mathsf{A} \vee \mathsf{B}$:

$\mathsf{A} \wedge \mathsf{B} \vdash \mathsf{A} \wedge \mathsf{B}$	$(Assume_P)$	(1)
$\mathsf{A} \wedge \mathsf{B} \vdash \mathsf{A}$	($\wedge EL$ on 1)	(2)
$\mathsf{A} \wedge \mathsf{B} \vdash \mathsf{A} \vee \mathsf{B}$	($\vee I$ on 2)	(3)

Notice that this is just a sequence where each line is either in $Assume_P$, or following from previous lines by applications of the rules. In other words, we have just written down a witnessing sequence in our generating system.

Definition 3.4.4. *A* deduction *is a witnessing sequence in $(Line_P, Assume_P, \mathcal{H})$.*

In other words, a deduction is a kind of "formal proof", where each step is governed by a limited collection of simple syntactic manipulations. Here is an an example of deduction showing that $\neg\mathsf{A}, \mathsf{A} \vee \mathsf{B} \vdash \mathsf{B}$. Notice that our deduction here sometimes adds and loses assumptions as it progresses:

$\neg\mathsf{A}, \mathsf{A}, \neg\mathsf{B} \vdash \mathsf{A}$	$(Assume_P)$	(1)
$\neg\mathsf{A}, \mathsf{A}, \neg\mathsf{B} \vdash \neg\mathsf{A}$	$(Assume_P)$	(2)
$\neg\mathsf{A}, \mathsf{A} \vdash \mathsf{B}$	($Contr$ on 1 and 2)	(3)
$\neg\mathsf{A}, \mathsf{B} \vdash \mathsf{B}$	$(Assume_P)$	(4)
$\neg\mathsf{A}, \mathsf{A} \vee \mathsf{B} \vdash \mathsf{B}$	($\vee PC$ on 3 and 4)	(5)

We are now ready to define a syntactic analogue to our our semantic notion $\Gamma \vDash \varphi$.

Definition 3.4.5. *Let P be a set, let $\Gamma \subseteq Form_P$, and let $\varphi \in Form_P$. We write $\Gamma \vdash \varphi$ if there exists a finite sequence $S \in \Gamma^*$ such that $S \vdash \varphi$. We pronounce $\Gamma \vdash \varphi$ as "Γ syntactically implies φ".*

Notice the slight distinction between Γ and S in this definition. We are using Γ to denote a *set* of formulas, and it's certainly possible that the set Γ is infinite. In contrast, S is a finite sequence of formulas from Γ. In particular, S is an ordered collection and allows repetition. More importantly, S must be finite. We enforce this condition because we want our formal proofs to be completely finite, and to follow simple syntactic rules which can be mechanically checked. We showed above that $\neg\mathsf{A}, \mathsf{A} \vee \mathsf{B} \vdash \mathsf{B}$, and hence we can conclude that $\{\neg\mathsf{A}, \mathsf{A} \vee \mathsf{B}\} \vdash \mathsf{B}$. If we have a countably infinite set $P = \{\mathsf{A}_1, \mathsf{A}_2, \mathsf{A}_3, \dots\}$, we still have that $\{\mathsf{A}_1, \mathsf{A}_2, \mathsf{A}_3, \dots\} \vdash \mathsf{A}_1 \wedge \mathsf{A}_2$ because we have $\mathsf{A}_1, \mathsf{A}_2 \vdash \mathsf{A}_1 \wedge \mathsf{A}_2$ (via a simple 3-line deduction).

For a longer example, but reasonably straightforward, example, we give a deduction showing that $\{\mathsf{A} \vee (\mathsf{B} \wedge \mathsf{C}) \vdash (\mathsf{A} \vee \mathsf{B}) \wedge (\mathsf{A} \vee \mathsf{C})$:

$\mathsf{A} \vdash \mathsf{A}$	$(Assume_P)$	(1)
$\mathsf{A} \vdash \mathsf{A} \vee \mathsf{B}$	$(\vee IL$ on 1)	(2)
$\mathsf{A} \vdash \mathsf{A} \vee \mathsf{C}$	$(\vee IL$ on 1)	(3)
$\mathsf{A} \vdash (\mathsf{A} \vee \mathsf{B}) \wedge (\mathsf{A} \vee \mathsf{C})$	$(\wedge I$ on 2 and 3)	(4)
$\mathsf{B} \wedge \mathsf{C} \vdash \mathsf{B} \wedge \mathsf{C}$	$(Assume_P)$	(5)
$\mathsf{B} \wedge \mathsf{C} \vdash \mathsf{B}$	$(\wedge EL$ on 5)	(6)
$\mathsf{B} \wedge \mathsf{C} \vdash \mathsf{A} \vee \mathsf{B}$	$(\vee IR$ on 6)	(7)
$\mathsf{B} \wedge \mathsf{C} \vdash \mathsf{C}$	$(\wedge ER$ on 5)	(8)
$\mathsf{B} \wedge \mathsf{C} \vdash \mathsf{A} \vee \mathsf{C}$	$(\vee IR$ on 8)	(9)
$\mathsf{B} \wedge \mathsf{C} \vdash (\mathsf{A} \vee \mathsf{B}) \wedge (\mathsf{A} \vee \mathsf{C})$	$(\wedge I$ on 7 and 9)	(10)
$\mathsf{A} \vee (\mathsf{B} \wedge \mathsf{C}) \vdash (\mathsf{A} \vee \mathsf{B}) \wedge (\mathsf{A} \vee \mathsf{C})$	$(\vee PC$ on 4 and 10)	(11)

We can also give deductions showing that $\Gamma \vdash \varphi$, even if we don't have concrete formulas. For example, our previous deduction showing that $\{\neg\mathsf{A}, \mathsf{A} \vee \mathsf{B}\} \vdash \mathsf{B}$ can be generalized to the following result.

Proposition 3.4.6. *For any set P and any $\varphi, \psi \in Form_P$, we have $\{\neg\varphi, \varphi \vee \psi\} \vdash \psi$.*

Proof. Let $\varphi, \psi \in Form_P$ be arbitrary. We give a deduction.

$\neg\varphi, \varphi, \neg\psi \vdash \varphi$	$(Assume_P)$	(1)
$\neg\varphi, \varphi, \neg\psi \vdash \neg\varphi$	$(Assume_P)$	(2)
$\neg\varphi, \varphi \vdash \psi$	$(Contr$ on 1 and 2)	(3)
$\neg\varphi, \psi \vdash \psi$	$(Assume_P)$	(4)
$\neg\varphi, \varphi \vee \psi \vdash \psi$	$(\vee PC$ on 3 and 4)	(5)

Therefore, $\{\neg\varphi, \varphi \vee \psi\} \vdash \psi$. □

For a more complicated example, consider the following deduction showing that $\{(\neg\varphi) \vee (\neg\psi)\} \vdash \neg(\varphi \wedge \psi)$ for all $\varphi, \psi \in Form_P$:

$\neg\varphi, \neg\neg(\varphi \wedge \psi), \neg(\varphi \wedge \psi) \vdash \neg(\varphi \wedge \psi)$	$(Assume_P)$	(1)
$\neg\varphi, \neg\neg(\varphi \wedge \psi), \neg(\varphi \wedge \psi) \vdash \neg\neg(\varphi \wedge \psi)$	$(Assume_P)$	(2)
$\neg\varphi, \neg\neg(\varphi \wedge \psi) \vdash \varphi \wedge \psi$	$(Contr$ on 1 and 2)	(3)
$\neg\varphi, \neg\neg(\varphi \wedge \psi) \vdash \varphi$	$(\wedge EL$ on 3)	(4)
$\neg\varphi, \neg\neg(\varphi \wedge \psi) \vdash \neg\varphi$	$(Assume_P)$	(5)
$\neg\varphi \vdash \neg(\varphi \wedge \psi)$	$(Contr$ on 4 and 5)	(6)
$\neg\psi, \neg\neg(\varphi \wedge \psi), \neg(\varphi \wedge \psi) \vdash \neg(\varphi \wedge \psi)$	$(Assume_P)$	(7)
$\neg\psi, \neg\neg(\varphi \wedge \psi), \neg(\varphi \wedge \psi) \vdash \neg\neg(\varphi \wedge \psi)$	$(Assume_P)$	(8)
$\neg\psi, \neg\neg(\varphi \wedge \psi) \vdash \varphi \wedge \psi$	$(Contr$ on 7 and 8)	(9)
$\neg\psi, \neg\neg(\varphi \wedge \psi) \vdash \psi$	$(\wedge ER$ on 9)	(10)
$\neg\psi, \neg\neg(\varphi \wedge \psi) \vdash \neg\psi$	$(Assume_P)$	(11)
$\neg\psi \vdash \neg(\varphi \wedge \psi)$	$(Contr$ on 10 and 11)	(12)
$(\neg\varphi) \vee (\neg\psi) \vdash \neg(\varphi \wedge \psi)$	$(\vee PC$ on 6 and 12)	(13)

We next show that $\vdash \varphi \vee \neg\varphi$ (i.e. that $\emptyset \vdash \varphi \vee \neg\varphi$) for all $\varphi \in Form_P$, illustrating how we can obtain a conclusion with an empty sequence of assumptions:

$$\begin{array}{rrl} \varphi \vdash \varphi & (Assume_P) & (1) \\ \varphi \vdash \varphi \vee \neg\varphi & (\vee IL \text{ on } 1) & (2) \\ \neg\varphi \vdash \neg\varphi & (Assume_P) & (3) \\ \neg\varphi \vdash \varphi \vee \neg\varphi & (\vee IR \text{ on } 3) & (4) \\ \lambda \vdash \varphi \vee \neg\varphi & (\neg PC \text{ on } 2 \text{ and } 4) & (5) \end{array}$$

Just as we defined a semantic way to say that set $\Gamma \subseteq Form_P$ is not contradictory (see our definition of satisfiable), we now define a syntactic counterpart.

Definition 3.4.7. *Γ is* inconsistent *if there exists $\theta \in Form_P$ such that $\Gamma \vdash \theta$ and $\Gamma \vdash \neg\theta$. Otherwise, we say that Γ is* consistent*.*

With all of these definitions and simple examples in hand, we can start to prove some simple results *about* the relation $\vdash$.

Proposition 3.4.8. *Let $\Gamma_1 \subseteq Form_P$ and $\Gamma_2 \subseteq Form_P$ be such that $\Gamma_1 \subseteq \Gamma_2$. If $\varphi \in Form_P$ is such that $\Gamma_1 \vdash \varphi$, then $\Gamma_2 \vdash \varphi$.*

Proof. Suppose that $\Gamma_1 \vdash \varphi$. We can then fix $S \in \Gamma_1^*$ with $S \vdash \varphi$. Since $S \in \Gamma_1^*$ and $\Gamma_1 \subseteq \Gamma_2$, we have that $S \in \Gamma_2^*$. Therefore, $\Gamma_2 \vdash \varphi$. □

Proposition 3.4.9. *Suppose that $S \in Form_P^*$ and $\varphi \in Form_P$ are such that $S \vdash \varphi$. If T is any permutation of S, then $T \vdash \varphi$.*

Proof. Using the *Reorder* rule, we can repeatedly flip adjacent elements of S until we form T. More formally, we are using the fact that the set of transpositions

$$\{(1\ 2), (2\ 3), (3\ 4), \ldots, (n-1\ \ n)\}$$

generates the symmetric group on n symbols. □

Proposition 3.4.10. *If $\Gamma \subseteq Form_P$ is inconsistent, then $\Gamma \vdash \varphi$ for all $\varphi \in Form_P$.*

Proof. Suppose that Γ is inconsistent. Fix $\theta \in Form_P$ with both $\Gamma \vdash \theta$ and $\Gamma \vdash \neg\theta$. By definition, we can then fix $S, T \in \Gamma^*$ with both $S \vdash \theta$ and $T \vdash \neg\theta$. Using the *Expand* rule together with Proposition 3.4.9, we conclude that $ST \vdash \theta$ and $ST \vdash \neg\theta$, where ST is the result of concatenating the sequences S and T.

Now let $\varphi \in Form_P$ be arbitrary. By the *Expand* rule, we have that $ST, \neg\varphi \vdash \theta$ and $ST, \neg\varphi \vdash \neg\theta$ (where the notation just means that we are concatenating the one element sequence $\neg\theta$ onto the end of ST). Using the *Contr* rule, we conclude that $ST \vdash \varphi$. Since $ST \in \Gamma^*$, it follows that $\Gamma \vdash \varphi$. □

Proposition 3.4.11. *Let $\Gamma \subseteq Form_P$ and let $\varphi \in Form_P$.*

1. *If $\Gamma \cup \{\varphi\}$ is inconsistent, then $\Gamma \vdash \neg\varphi$.*
2. *If $\Gamma \cup \{\neg\varphi\}$ is inconsistent, then $\Gamma \vdash \varphi$.*

Proof.

1. Suppose that $\Gamma \cup \{\varphi\}$ is inconsistent. By Proposition 3.4.10, we then have that $\Gamma \cup \{\varphi\} \vdash \neg\varphi$. Fix $S \in (\Gamma \cup \{\varphi\})^*$ such that $S \vdash \neg\varphi$. By the *Expand* rule, we then have $S, \varphi \vdash \neg\varphi$. Using Proposition 3.4.9 and the *Delete* rule, we can fix $T \in \Gamma^*$ with $T, \varphi \vdash \neg\varphi$. Now we also trivially have $T, \neg\varphi \vdash \neg\varphi$ from $Assume_P$. Using the $\neg PC$ rule, it follows that $T \vdash \neg\varphi$. Since $T \in \Gamma^*$, we conclude that $\Gamma \vdash \neg\varphi$.

2. Suppose that $\Gamma \cup \{\neg\varphi\}$ is inconsistent. By Proposition 3.4.10, we then have that $\Gamma \cup \{\neg\varphi\} \vdash \varphi$. Fix $S \in (\Gamma \cup \{\neg\varphi\})^*$ such that $S \vdash \varphi$. By the *Expand* rule, we then have $S, \neg\varphi \vdash \varphi$. Using Proposition 3.4.9 and the *Delete* rule, we can fix $T \in \Gamma^*$ with $T, \neg\varphi \vdash \varphi$. Now we also trivially have $T, \varphi \vdash \varphi$ from $Assume_P$. Using the $\neg PC$ rule, it follows that $T \vdash \varphi$. Since $T \in \Gamma^*$, we conclude that $\Gamma \vdash \varphi$.

□

Corollary 3.4.12. *Let $\varphi \in Form_P$. If $\Gamma \subseteq Form_P$ is consistent, then either $\Gamma \cup \{\varphi\}$ is consistent or $\Gamma \cup \{\neg\varphi\}$ is consistent.*

Proof. We prove the contrapositive. If both $\Gamma \cup \{\varphi\}$ and $\Gamma \cup \{\neg\varphi\}$ are inconsistent, then both $\Gamma \vdash \neg\varphi$ and $\Gamma \vdash \varphi$ by Proposition 3.4.11, so Γ is inconsistent. □

Proposition 3.4.13. *Let $\Gamma \subseteq Form_P$ and let $\varphi \in Form_P$.*

1. *If $\Gamma \vdash \varphi$ and $\Gamma \cup \{\varphi\} \vdash \psi$, then $\Gamma \vdash \psi$.*

2. *If $\Gamma \vdash \varphi$ and $\Gamma \vdash \varphi \to \psi$, then $\Gamma \vdash \psi$.*

Proof.

1. Suppose that $\Gamma \vdash \varphi$ and $\Gamma \cup \{\varphi\} \vdash \psi$. Using Proposition 3.4.8, we conclude that $\Gamma \cup \{\neg\varphi\} \vdash \varphi$. Now we also trivially have $\Gamma \cup \{\neg\varphi\} \vdash \neg\varphi$, so Γ is inconsistent. Using Proposition 3.4.10, it follows that $\Gamma \cup \{\neg\varphi\} \vdash \psi$. By definition, together with the *Expand* rule, the *Delete* rule, and Proposition 3.4.9, we can fix $S \in \Gamma^*$ such that $S, \neg\varphi \vdash \psi$. Similarly, using the fact that $\Gamma \cup \{\varphi\} \vdash \psi$, we can fix $T \in \Gamma^*$ such that $T, \varphi \vdash \psi$. By using the *Expand* rule and Proposition 3.4.9 again, we have both $ST, \neg\varphi \vdash \psi$ and $ST, \varphi \vdash \psi$. Applying the $\neg PC$ rule, we conclude that $ST \vdash \varphi$. Since $ST \in \Gamma^*$, it follows that $\Gamma \vdash \psi$.

2. Suppose that $\Gamma \vdash \varphi$ and $\Gamma \vdash \varphi \to \psi$. Fix $S \in \Gamma$ with $S \vdash \varphi \to \psi$. Using the $\to E$ rule, we have $S, \varphi \vdash \psi$. Since $S \in \Gamma^*$, it follows that $\Gamma \cup \{\varphi\} \vdash \psi$. Now apply (1).

□

Since deductions are defined entirely in terms of finite sequences, we also have the following simple result.

Proposition 3.4.14. *$\Gamma \vdash \varphi$ if and only if there is a finite $\Gamma_0 \subseteq \Gamma$ such that $\Gamma_0 \vdash \varphi$.*

Proof. The right-to-left direction is immediate from Proposition 3.4.8. For the left-to-right direction, suppose that $\Gamma \vdash \varphi$. Fix $S \in \Gamma^*$ such that $S \vdash \varphi$. Letting Γ_0 be the finite subset of Γ consisting of those element that occur in S, we immediately conclude that $\Gamma_0 \vdash \varphi$. □

Corollary 3.4.15. *If every finite subset of Γ is consistent, then Γ is consistent.*

Proof. We prove the contrapositive. Suppose that Γ is inconsistent, and fix $\theta \in Form_P$ such that $\Gamma \vdash \theta$ and $\Gamma \vdash \neg\theta$. By Proposition 3.4.14, there exists finite sets $\Gamma_0 \subseteq \Gamma$ and $\Gamma_1 \subseteq \Gamma$ such that $\Gamma_0 \vdash \theta$ and $\Gamma_1 \vdash \neg\theta$. Using Proposition 3.4.8, it follows that $\Gamma_0 \cup \Gamma_1 \vdash \theta$ and $\Gamma_0 \cup \Gamma_1 \vdash \neg\theta$, so $\Gamma_0 \cup \Gamma_1$ is a finite inconsistent subset of Γ. □

3.5 Soundness and Completeness

We now have two notions of implications: the semantic $\Gamma \vDash \varphi$ and the syntactic $\Gamma \vdash \varphi$. We also have two ways to say that a set of formulas is not contradictory: the semantic notion of satisfiability and the syntactic notion of consistency. Although these concepts are defined in very different ways, it turns out that the semantic and syntactic notions are the same in both cases. One direction of each of these equivalences is known and the Soundness Theorem, and the other direction is known as the Completeness Theorem.

The heart of the Soundness Theorem is the statement that if $\Gamma \vdash \varphi$, then $\Gamma \vDash \varphi$. Intuitively, if we have a formal proof of a statement, then whenever we assign true/false values in a way that makes the assumptions true, we should expect that the conclusion is also true. More formally, we need need to argue that if we have a deduction witnessing that $\Gamma \vdash \varphi$, and we have a truth assignment $M \colon P \to \{0,1\}$ with the property that $v_M(\gamma) = 1$ for all $\gamma \in \Gamma$, then we must have $v_M(\varphi) = 1$. A deduction is just a finite sequence of steps, and a deduction showing that $\Gamma \vdash \varphi$ ends with a line $T \vdash \varphi$ for some $T \in \Gamma^*$. Now in order to show that $\Gamma \vDash \varphi$, it suffices to show that whenever we have a truth assignment $M \colon P \to \{0,1\}$ with the property that $v_M(\gamma) = 1$ for all γ that appear in T, then we must have $v_M(\varphi) = 1$. Rather than deal with the last line of the deduction directly, it is much easier to work through the deduction from beginning to end, and argue that each line has this property. To do that, what we really want to show is that the elements of $Assume_P$ have this property, and that each of our proof rules *preserve* this property. In other words, we want to given an inductive argument on the generating set used to define syntactic implication. In order to carry out this argument, we extend our notation for $\vDash$ to allow finite sequences of formulas.

Notation 3.5.1. *Given $S \in Form_P^*$ and $\varphi \in Form_P$, we write $S \vDash \varphi$ to mean that whenever M is a truth assignment on P with the property that $v_M(\gamma) = 1$ for all γ that appear in S, we have $v_M(\varphi) = 1$. In other words, whenever M is a truth assignment on P with the property that $v_M(S(i)) = 1$ for all $i < |S|$, we have $v_M(\varphi) = 1$.*

Now we are ready to state and prove the Soundness Theorem. As alluded to above, instead of working with deductions directly and thinking about going line by line, it is more elegant to organize the argument as induction on the generating system tied to syntactic implication.

Theorem 3.5.2 (Soundness Theorem)**.** *Let P be a set.*

1. *If $\Gamma \vdash \varphi$, then $\Gamma \vDash \varphi$.*
2. *Every satisfiable set of formulas is consistent.*

Proof.

1. We prove the following fact: If $S \in Form_P^*$ and $\varphi \in Form_P$ are such that $S \vdash \varphi$, then $S \vDash \varphi$. To see why this suffices, suppose that $\Gamma \vdash \varphi$. By definition, we can then fix $S \in \Gamma^*$ with $S \vdash \varphi$. From here we can conclude that $S \vDash \varphi$. Since every element of S is an element of Γ, it follows that $\Gamma \vDash \varphi$.

   We prove the statement "Whenever $S \vdash \varphi$, we have $S \vDash \varphi$" by induction. In other words, if G is the set generated by starting with $Assume_P$ and using our proof rules, and we let

   $$X = \{(S, \varphi) \in G : S \vDash \varphi\},$$

   then we show by induction on G that $X = G$. We begin by noting that if φ appears in the sequence S, then we trivially have $S \vDash \varphi$ by definition. Therefore, $(S, \varphi) \in X$ for all $(S, \varphi) \in Assume_P$. We now handle the inductive steps, one for each rule.

   - We first handle the $\wedge EL$ rule. Suppose that $S \vDash \varphi \wedge \psi$. We need to show that $S \vDash \varphi$. However, this is straightforward because if $M \colon P \to \{0,1\}$ is such that $v_M(\gamma) = 1$ for all γ appearing in S, then $v_M(\varphi \wedge \psi) = 1$ because $S \vDash \varphi \wedge \psi$, hence $v_M(\varphi) = 1$. Therefore, $S \vDash \varphi$. The other $\wedge$ rules and the $\vee$ rules are similar.

- Consider $\to E$ rule. Suppose that $S \vDash \varphi \to \psi$. We need to show that $S, \varphi \vDash \psi$. Let $M\colon P \to \{0,1\}$ be such that $v_M(\gamma) = 1$ for all γ appearing in S, φ. Since $S \vDash \varphi \to \psi$, we have $v_M(\varphi \to \psi) = 1$. Since $v_M(\varphi) = 1$, it follows that we must have $v_M(\psi) = 1$. Therefore, $S, \varphi \vDash \psi$. The $\to I$ rule is similar.
- We now handle the $\neg PC$ rule. Suppose that $S, \psi \vDash \varphi$ and $S, \neg\psi \vDash \varphi$. We need to show that $S \vDash \varphi$. Let $M\colon P \to \{0,1\}$ be such that $v_M(\gamma) = 1$ for all γ appearing in S. Now either $v_M(\psi) = 1$ or $v_M(\psi) = 0$. If $v_M(\psi) = 1$, then we must have $v_M(\varphi) = 1$ because $S, \psi \vDash \varphi$. Otherwise, we have $v_M(\psi) = 0$, hence $v_M(\neg\psi) = 1$, and thus $v_M(\varphi) = 1$ because $S, \neg\psi \vDash \varphi$. Therefore, $S \vDash \varphi$. The $\vee PC$ rule is similar.
- Consider the *Contr* rule. Suppose that $S, \neg\varphi \vDash \psi$ and $S, \neg\varphi \vDash \neg\psi$. We need to show that $S \vDash \varphi$. Let $M\colon P \to \{0,1\}$ be such that $v_M(\gamma) = 1$ for all γ appearing in S. Suppose instead that $v_M(\varphi) = 0$. We then have $v_M(\neg\varphi) = 1$, and using the fact that $S, \neg\varphi \vDash \psi$ and $S, \neg\varphi \vDash \neg\psi$, we conclude that both $v_M(\psi) = 1$ and $v_M(\neg\psi) = 1$. This is a contradiction, hence we must have $v_M(\varphi) = 1$. Therefore, $S \vDash \varphi$.
- The assumption transformation rules are all completely straightforward.

By induction, it follows that $X = G$, which is to say that whenever $S \vdash \varphi$, we have $S \vDash \varphi$. By the comments above, statement (1) follows

2. Let Γ be a satisfiable set of formulas. Fix a truth assignment $M\colon P \to \{0,1\}$ such that $v_M(\gamma) = 1$ for all $\gamma \in \Gamma$. Suppose instead that Γ is inconsistent, and fix $\theta \in Form_P$ such that $\Gamma \vdash \theta$ and $\Gamma \vdash \neg\theta$. We then have $\Gamma \vDash \theta$ and $\Gamma \vDash \neg\theta$ by part (1), hence both $v_M(\theta) = 1$ and $v_M(\neg\theta) = 1$, a contradiction. It follows that Γ must be consistent.

□

The Completeness Theorem is the converse (of both parts) of the Soundness Theorem. In order words, it says that (1) If $\Gamma \vDash \varphi$, then $\Gamma \vdash \varphi$ and (2) every consistent set of formulas is satisfiable. Part (1) looks quite difficult to tackle directly (think about the amount of cleverness that went into finding the simple deductions above), so instead we go after (2) first and then use it to prove (1).

Assume then that we have a consistent set of formulas $\Gamma \subseteq Form_P$. We need to build a truth assignment $M\colon P \to \{0,1\}$ such that $v_M(\gamma) = 1$ for all $\gamma \in \Gamma$. Suppose that we are trying to define $M(\mathsf{A})$ for a given $\mathsf{A} \in P$. If $\mathsf{A} \in \Gamma$, then we should certainly set $M(\mathsf{A}) = 1$. Similarly, if $\neg\mathsf{A} \in \Gamma$, then we should set $M(\mathsf{A}) = 0$. But what should we do if both $\mathsf{A} \notin \Gamma$ and $\neg\mathsf{A} \notin \Gamma$? What if every formula in Γ is very long and complex, so we have no idea how to start defining the truth assignment? The idea is to expand Γ to a larger consistent set which has some "simpler" formulas that aid us in deciphering how to define M. Ideally, we would like to extend Γ to consistent set Γ' such that for all $\mathsf{A} \in P$, either $\mathsf{A} \in \Gamma'$ or $\neg\mathsf{A} \in \Gamma'$, because that would give us a clear way to define M. Once we have such a natural way to define M, we would then have to verify that $v_M(\gamma) = 1$ for all $\gamma \in \Gamma'$. In order to "move up" from the literals to more complicated formulas in Γ', we would prefer to have intermediate formulas along the way so that we can keep track of what is happening, which will aid an inductive argument. To this end, we generalize the idea of having either A or $\neg\mathsf{A}$ appear in our set to the following.

Definition 3.5.3. *Let $\Delta \subseteq Form_P$. We say that Δ is* complete *if for all $\varphi \in Form_P$, either $\varphi \in \Delta$ or $\neg\varphi \in \Delta$.*

Our first task is to show that if Γ is consistent, then it can be expanded to a consistent and complete set Δ. We begin by proving this in the special case when P is countable because the construction is more transparent and avoids more powerful set-theoretic tools.

Proposition 3.5.4. *Suppose that P is countable. If Γ is consistent, then there exists a set $\Delta \supseteq \Gamma$ which is consistent and complete.*

Proof. Since P is countable, we have that $Sym_P = P \cup \{\neg, \wedge, \vee, \rightarrow\}$ is countable, and therefore Sym_P^* is countable by Corollary A.2.4. Since $Form_P \subseteq Sym_P^*$, it follows that $Form_P$ is countable. Alternatively, we could use apply Problem 6 on Homework 1 to conclude that $Form_P$ is countable.

Since $Form_P$ is countable, we can list $Form_P$ as $\psi_1, \psi_2, \psi_3, \ldots$ (formally, we are fixing a surjection $f\colon \mathbb{N}^+ \rightarrow Form_P$). We define a sequence of sets $\Gamma_0, \Gamma_1, \Gamma_2, \ldots$ recursively as follows. We begin by letting $\Gamma_0 = \Gamma$. Suppose that $n \in \mathbb{N}$ and we that we have defined Γ_n. Let

$$\Gamma_{n+1} = \begin{cases} \Gamma_n \cup \{\psi_n\} & \text{if } \Gamma_n \cup \{\psi_n\} \text{ is consistent} \\ \Gamma_n \cup \{\neg\psi_n\} & \text{otherwise.} \end{cases}$$

Using induction on $\mathbb{N}$ together with Corollary 3.4.12, it follows that Γ_n is consistent for all $n \in \mathbb{N}$. Let $\Delta = \bigcup_{n\in\mathbb{N}} \Gamma_n$.

We first argue that Δ is consistent. For any finite subset Δ_0 of Δ, there exists an $n \in \mathbb{N}$ such that $\Delta_0 \subseteq \Gamma_n$, and so Δ_0 is consistent because every Γ_n is consistent (here we are using the fact that a finite sequence from Δ_0 is a finite sequence from some Γ_n). Therefore, Δ is consistent by Corollary 3.4.15. We end by arguing that Δ is complete. Let $\varphi \in Form_P$ be arbitrary, and fix $n \in \mathbb{N}^+$ such that $\varphi = \psi_n$. By construction, we either have $\varphi \in \Gamma_{n+1} \subseteq \Delta$ or $\neg\varphi \in \Gamma_{n+1} \subseteq \Delta$. Therefore, Δ is complete. □

How can we handle the case where P is uncountable? Intuitively, we want to allow the listing of the formulas to continue "beyond" finite stages, and we will eventually develop the tools of transfinite induction and recursion to accomplish such awe-inspiring tasks. Another approach is to invoke a useful tool known as Zorn's Lemma (which is really a transfinite recursion in disguise, as we will eventually see). The idea is that a complete consistent set is just a maximal consistent set, where *maximal* means that it is not strictly contained in any other consistent set. The connection here is that Zorn's Lemma allows us to prove the existence of maximal elements in certain partial orderings. If you are unfamiliar with Zorn's Lemma, feel free to focus only on the countable case until we cover set theory.

Definition 3.5.5. *Δ is* maximal consistent *if Δ is consistent and there is no $\Delta' \supset \Delta$ which is consistent.*

Proposition 3.5.6. *Let $\Delta \subseteq Form_P$. Δ is maximal consistent if and only if Δ is consistent and complete.*

Proof. Suppose that Δ is maximal consistent. We certainly have that Δ is consistent. Let $\varphi \in Form_P$ be arbitrary. By Corollary 3.4.12, either $\Delta \cup \{\varphi\}$ is consistent or $\Delta \cup \{\neg\varphi\}$ is consistent. If $\Delta \cup \{\varphi\}$ is consistent, then $\varphi \in \Delta$ because Δ is maximal consistent. Similarly, If $\Delta \cup \{\neg\varphi\}$ is consistent, then $\neg\varphi \in \Delta$ because Δ is maximal consistent. Therefore, either $\varphi \in \Delta$ or $\neg\varphi \in \Delta$.

Suppose conversely that Δ is consistent and complete. Let $\Delta' \supset \Delta$ be arbitrary and fix $\varphi \in \Delta' \backslash \Delta$. Since Δ is complete and $\varphi \notin \Delta$, we have $\neg\varphi \in \Delta$. Since we have both $\varphi, \neg\varphi \in \Delta'$, we trivially have both $\Delta' \vdash \varphi$ and $\Delta' \vdash \neg\varphi$, so Δ' is inconsistent. It follows that Δ is maximal consistent. □

Proposition 3.5.7. *If $\Gamma \subseteq Form_P$ is consistent, then there exists a set $\Delta \supseteq \Gamma$ which is consistent and complete.*

Proof. Consider an arbitrary consistent $\Gamma \subseteq Form_P$. Let $\mathcal{Q} = \{\Phi \subseteq Form_P : \Gamma \subseteq \Phi \text{ and } \Phi \text{ is consistent}\}$, and order $\mathcal{Q}$ by $\subseteq$. Notice that $\mathcal{Q}$ is nonempty because $\Gamma \in \mathcal{Q}$. Let $\mathcal{C} \subseteq \mathcal{Q}$ be an arbitrary chain in $\mathcal{Q}$. Let $\Psi = \bigcup \mathcal{C} = \{\psi \in Form_P : \psi \in \Phi \text{ for some } \Phi \in \mathcal{C}\}$. We need to argue that Ψ is consistent. Suppose that Ψ_0 is a finite subset of Ψ, say $\Psi_0 = \{\psi_1, \psi_2, \ldots, \psi_n\}$. For each ψ_i, fix $\Phi_i \in \mathcal{C}$ with $\psi_i \in \Phi_i$. Since $\mathcal{C}$ is a chain, there exists j such that $\Phi_j \supseteq \Phi_i$ for all i. Now $\Phi_j \in \mathcal{C} \subseteq \mathcal{Q}$, so Φ_j is consistent, and hence Ψ_0 is consistent. Therefore, Ψ is consistent by Corollary 3.4.15. It follows that $\Psi \in \mathcal{Q}$ and using the fact that $\Phi \subseteq \Psi$ for all $\Phi \in \mathcal{C}$, we may conclude that $\mathcal{C}$ has an upper bound.

Therefore, by Zorn's Lemma, $\mathcal{Q}$ has a maximal element Δ. Notice that Δ is maximal consistent, hence Δ is complete and consistent by Proposition 3.5.6. □

Lemma 3.5.8. *Let $\Delta \subseteq Form_P$ be consistent and complete, and let $\varphi \in Form_P$. If $\Delta \vdash \varphi$, then $\varphi \in \Delta$.*

Proof. Suppose that $\Delta \vdash \varphi$. Since Δ is complete, we have that either $\varphi \in \Delta$ or $\neg\varphi \in \Delta$. Now if $\neg\varphi \in \Delta$, then we would would trivially have $\Delta \vdash \neg\varphi$ (in addition to our assumed $\Delta \vdash \varphi$), contradicting the fact that Δ is consistent. It follows that $\varphi \in \Delta$. □

Suppose now that we have a consistent and complete $\Delta \subseteq Form_P$. For each $\mathsf{A} \in P$, we then have that either $\mathsf{A} \in \Delta$ or $\neg\mathsf{A} \in \Delta$, but not both. As mentioned above, this provides us with a natural truth assignment $M\colon P \to \{0,1\}$ that will make all of the literals in Δ true. Now we need to argue that the rest of the formulas in Δ are true under M, and the following lemma is the key inductive "glue" that we will use to work our up through more complicated formulas.

Lemma 3.5.9. *Suppose that Δ is consistent and complete. We have the following:*

1. *$\neg\varphi \in \Delta$ if and only if $\varphi \notin \Delta$.*
2. *$\varphi \wedge \psi \in \Delta$ if and only if $\varphi \in \Delta$ and $\psi \in \Delta$.*
3. *$\varphi \vee \psi \in \Delta$ if and only if either $\varphi \in \Delta$ or $\psi \in \Delta$.*
4. *$\varphi \to \psi \in \Delta$ if and only if either $\varphi \notin \Delta$ or $\psi \in \Delta$.*

Proof.

1. Suppose first that $\neg\varphi \in \Delta$. Now if $\varphi \in \Delta$ as well, then we would have both $\Delta \vdash \neg\varphi$ and $\Delta \vdash \varphi$ trivially, contradicting the fact that Δ is consistent. It follows that $\varphi \notin \Delta$.

   Conversely, if $\varphi \notin \Delta$, then $\neg\varphi \in \Delta$ because Δ is complete.

2. Suppose first that $\varphi \wedge \psi \in \Delta$. Since

   $$\begin{array}{lll} \varphi \wedge \psi \vdash \varphi \wedge \psi & (Assume_P) & (1) \\ \varphi \wedge \psi \vdash \varphi & (\wedge EL \text{ on } 1) & (2) \end{array}$$

   and

   $$\begin{array}{lll} \varphi \wedge \psi \vdash \varphi \wedge \psi & (Assume_P) & (1) \\ \varphi \wedge \psi \vdash \psi & (\wedge ER \text{ on } 1) & (2) \end{array}$$

   are both deductions, and $\varphi \wedge \psi \in \Delta$, we have both $\Delta \vdash \varphi$ and $\Delta \vdash \psi$. Using Lemma 3.5.8, we conclude that both $\varphi \in \Delta$ and $\psi \in \Delta$.

   Conversely, suppose that $\varphi \in \Delta$ and $\psi \in \Delta$. Consider the following deduction:

   $$\begin{array}{lll} \varphi, \psi \vdash \varphi & (Assume_P) & (1) \\ \varphi, \psi \vdash \psi & (Assume_P) & (2) \\ \varphi, \psi \vdash \varphi \wedge \psi & (\vee I \text{ on } 2) & (3). \end{array}$$

   Since $\varphi, \psi \in \Delta$, we have $\Delta \vdash \varphi \wedge \psi$. Using Lemma 3.5.8, we conclude that $\varphi \wedge \psi \in \Delta$.

3. Suppose first that $\varphi \vee \psi \in \Delta$. If $\varphi \in \Delta$, then we are done, so assume that $\varphi \notin \Delta$. Since Δ is complete, we have that $\neg\varphi \in \Delta$. Now in Proposition 3.4.6, we showed that that $\neg\varphi, \varphi \vee \psi \vdash \psi$, so since we have both $\neg\varphi \in \Delta$ and $\varphi \vee \psi \in \Delta$, we conclude that $\Delta \vdash \psi$. Using Lemma 3.5.8, we conclude that $\psi \in \Delta$.

Conversely, suppose that either $\varphi \in \Delta$ or $\psi \in \Delta$. Since

$$\begin{array}{lll} \varphi \vdash \varphi & (Assume_P) & (1) \\ \varphi \vdash \varphi \vee \psi & (\vee IL \text{ on } 1) & (2) \end{array}$$

and

$$\begin{array}{lll} \psi \vdash \psi & (Assume_P) & (1) \\ \psi \vdash \varphi \vee \psi & (\vee IR \text{ on } 1) & (2) \end{array}$$

are both deductions, and we are assuming that either $\varphi \in \Delta$ or $\psi \in \Delta$, we conclude that either $\Delta \vdash \varphi$ or $\Delta \vdash \psi$. Using Lemma 3.5.8, we conclude that either $\varphi \in \Delta$ or $\psi \in \Delta$.

4. Suppose first that $\varphi \to \psi \in \Delta$. If $\varphi \notin \Delta$, then we are done, so assume that $\varphi \in \Delta$. Since

$$\begin{array}{lll} \varphi \to \psi \vdash \varphi \to \psi & (Assume_P) & (1) \\ \varphi \to \psi, \varphi \vdash \psi & (\to E \text{ on } 1) & (2) \end{array}$$

is a deduction, and both $\varphi \to \psi \in \Delta$ and $\varphi \in \Delta$, we have $\Delta \vdash \psi$. Using Lemma 3.5.8, we conclude that $\psi \in \Delta$.

Conversely, suppose that either $\varphi \notin \Delta$ or $\psi \in \Delta$.

Case 1: Suppose that $\varphi \notin \Delta$. Since Δ is complete, we then have $\neg\varphi \in \Delta$. Since

$$\begin{array}{lll} \neg\varphi, \varphi, \neg\psi \vdash \varphi & (Assume_P) & (1) \\ \neg\varphi, \varphi, \neg\psi \vdash \neg\varphi & (Assume_P) & (2) \\ \neg\varphi, \varphi \vdash \psi & (Contr \text{ on } 1 \text{ and } 2) & (3) \\ \neg\varphi \vdash \varphi \to \psi & (\to I \text{ on } 3) & (4) \end{array}$$

is a deduction, and $\neg\varphi \in \Delta$, we have $\Delta \vdash \varphi \to \psi$. Using Lemma 3.5.8, we conclude that $\varphi \to \psi \in \Delta$.

Case 2: Suppose that $\psi \in \Delta$. Since

$$\begin{array}{lll} \psi, \varphi \vdash \psi & (Assume_P) & (1) \\ \psi \vdash \varphi \to \psi & (\to I \text{ on } 1) & (2) \end{array}$$

is a deduction, and $\psi \in \Delta$, we have $\Delta \vdash \varphi \to \psi$. Using Lemma 3.5.8, we conclude that $\varphi \to \psi \in \Delta$.

Therefore, in either case, we have $\varphi \to \psi \in \Delta$.

□

Proposition 3.5.10. *If Δ is consistent and complete, then Δ is satisfiable.*

Proof. Suppose that Δ is complete and consistent. Define $M\colon P \to \{0, 1\}$ as follows:

$$M(\mathsf{A}) = \begin{cases} 1 & \text{if } \mathsf{A} \in \Delta \\ 0 & \text{if } \mathsf{A} \notin \Delta. \end{cases}$$

We prove by induction on φ that $\varphi \in \Delta$ if and only if $v_M(\varphi) = 1$. For any $\mathsf{A} \in P$, we have

$$\mathsf{A} \in \Delta \Leftrightarrow M(\mathsf{A}) = 1 \Leftrightarrow v_M(\mathsf{A}) = 1$$

by our definition of M.

Suppose that the statement is true for φ. We have

$$\begin{aligned} \neg\varphi \in \Delta &\Leftrightarrow \varphi \notin \Delta && \text{(by Lemma 3.5.9)} \\ &\Leftrightarrow v_M(\varphi) = 0 && \text{(by induction)} \\ &\Leftrightarrow v_M(\neg\varphi) = 1 \end{aligned}$$

Suppose that the statement is true for φ and ψ. We have

$$\begin{aligned} \varphi \wedge \psi \in \Delta &\Leftrightarrow \varphi \in \Delta \text{ and } \psi \in \Delta && \text{(by Lemma 3.5.9)} \\ &\Leftrightarrow v_M(\varphi) = 1 \text{ and } v_M(\psi) = 1 && \text{(by induction)} \\ &\Leftrightarrow v_M(\varphi \wedge \psi) = 1 \end{aligned}$$

and

$$\begin{aligned} \varphi \vee \psi \in \Delta &\Leftrightarrow \varphi \in \Delta \text{ or } \psi \in \Delta && \text{(by Lemma 3.5.9)} \\ &\Leftrightarrow v_M(\varphi) = 1 \text{ or } v_M(\psi) = 1 && \text{(by induction)} \\ &\Leftrightarrow v_M(\varphi \vee \psi) = 1 \end{aligned}$$

and finally

$$\begin{aligned} \varphi \to \psi \in \Delta &\Leftrightarrow \varphi \notin \Delta \text{ or } \psi \in \Delta && \text{(by Lemma 3.5.9)} \\ &\Leftrightarrow v_M(\varphi) = 0 \text{ or } v_M(\psi) = 1 && \text{(by induction)} \\ &\Leftrightarrow v_M(\varphi \to \psi) = 1 \end{aligned}$$

Therefore, by induction, we have $\varphi \in \Delta$ if and only if $v_M(\varphi) = 1$. In particular, we have $v_M(\varphi) = 1$ for all $\varphi \in \Delta$, hence Δ is satisfiable. □

We now have all of the ingredients in place to prove Completeness Theorem. We state it for an arbitrary set P, but recall that the uncountable case used Zorn's Lemma to extend to a complete and consistent set.

Theorem 3.5.11 (Completeness Theorem)**.** *Let P be a set.*

1. *Every consistent set of formulas is satisfiable.*

2. *If $\Gamma \vDash \varphi$, then $\Gamma \vdash \varphi$.*

Proof.

1. Suppose that Γ is consistent. By Proposition 3.5.7, we may fix $\Delta \supseteq \Gamma$ which is consistent and complete. Now Δ is satisfiable by Proposition 3.5.10, so we may fix $M\colon P \to \{0,1\}$ such that $v_M(\delta) = 1$ for all $\delta \in \Delta$. Since $\Gamma \subseteq \Delta$, it follows that $v_M(\gamma) = 1$ for all $\gamma \in \Gamma$. Therefore, Γ is satisfiable.

2. Suppose that $\Gamma \vDash \varphi$. We then have that $\Gamma \cup \{\neg\varphi\}$ is unsatisfiable, hence $\Gamma \cup \{\neg\varphi\}$ is inconsistent by part 1. Using Proposition 3.4.11, it follows that that $\Gamma \vdash \varphi$.

□

3.6 Compactness and Applications

We have done a lot of hard work to show that our semantic and syntactic definitions coincide. As a result, we now know that it is possible, at least in principle, to find all semantic consequences by following simple syntactic rules on finite sequences. The primary way that will take advantage of this fact is by using Proposition 3.4.14 and Corollary 3.4.15, which are formalizations of the intuition that any syntactic deduction can only make use of finitely many of the assumptions. Translating to the semantic side, we arrive at the following fundamental, and surprising, result.

Corollary 3.6.1 (Compactness Theorem)**.** *Let P be a set.*

1. *If $\Gamma \vDash \varphi$, then there exists a finite $\Gamma_0 \subseteq \Gamma$ such that $\Gamma_0 \vDash \varphi$.*
2. *If every finite subset of Γ is satisfiable, then Γ is satisfiable.*

Proof. We first prove 1. Suppose that $\Gamma \vDash \varphi$. By the Completeness Theorem, we have $\Gamma \vdash \varphi$. Using Proposition 3.4.14, we may fix a finite $\Gamma_0 \subseteq \Gamma$ such that $\Gamma_0 \vdash \varphi$. By the Soundness Theorem, we have $\Gamma_0 \vDash \varphi$.

We now prove 2. If every finite subset of Γ is satisfiable, then every finite subset of Γ is consistent by the Soundness Theorem, hence Γ is consistent by Corollary 3.4.15, and so Γ is satisfiable by the Completeness Theorem. □

We now show how to use the Compactness Theorem to prove mathematical results. We start with an example about graphs. For our purposes here, a *graph* is an ordered pair (V, E), where V is a set, and $E \subseteq V^2$ is a binary relation on V that is symmetric, i.e. whenever $(a, b) \in E$, we also have $(b, a) \in E$. In other words, instead of coding edges are (unordered) subsets of V of size 2, we simply code them as ordered pairs, and require that whenever we have an ordered pair, then we also have the reverse pair. If we wanted to define directed graphs in this way, we simply drop the symmetric assumption. Notice that our graphs do allow loops (since we could have $(a, a) \in E$), but do not permit multiple edges that have the same endpoints.

Given a graph G, an interesting problem in both mathematical modeling and computer science is to determine whether we can color the vertices of the graph (using a small number of colors), in such a way that adjacent vertices have distinct colorings. We begin by formally defining these vertex colorings.

Definition 3.6.2. *Let $G = (V, E)$ be a graph and let $k \in \mathbb{N}^+$.*

1. *A k-coloring of G is a function $f : V \to [k]$.*
2. *We say that a k-coloring f of G is* proper *if $f(u) \neq f(w)$ whenever $(u, w) \in E$.*
3. *We say that G is* k-colorable *if there exists a proper k-coloring of G.*

Proposition 3.6.3. *Let $G = (V, E)$ be a (possibly infinite) graph and let $k \in \mathbb{N}^+$. If every finite subgraph of G is k-colorable, then G is k-colorable.*

The idea is to introduce a propositional symbol $\mathsf{A}_{u,i}$ for each ordered pair consisting of of a vertex $u \in V$ and possible color $i \in [k]$. Intuitively, a truth assignment M with $M(\mathsf{A}_{u,i}) = 1$ is an instruction to color the vertex u with the color i. We then code the various requirements on a coloring into formulas. Here is the argument.

Proof. Let $P = \{\mathsf{A}_{u,i} : u \in V \text{ and } i \in [k]\}$, and let Γ be the union of the following sets:

- $\left\{ \bigvee_{i=0}^{k-1} \mathsf{A}_{u,i} : u \in V \right\}$.
- $\{\neg(\mathsf{A}_{u,i} \wedge \mathsf{A}_{u,j}) : u \in V \text{ and } i, j \in [k] \text{ with } i \neq j\}$.

- $\{\neg(\mathsf{A}_{u,i} \wedge \mathsf{A}_{w,i}) : (u,w) \in E \text{ and } i \in [k]\}$.

We use the Compactness Theorem to show that Γ is satisfiable. Let $\Gamma_0 \subseteq \Gamma$ be an arbitrary finite subset of Γ. Let $\{u_1, u_2, \ldots, u_n\}$ be all of the elements $u \in V$ such that $\mathsf{A}_{u,i}$ occurs in some element of Γ_0 for some i. Since every finite subgraph of G is k-colorable, we may fix a proper k-coloring $f\colon \{u_1, u_2, \ldots, u_n\} \to [k]$ of the subgraph of G induced by $\{u_1, u_2, \ldots, u_n\}$. If we define a truth assignment $M\colon P \to \{0,1\}$ by

$$M(\mathsf{A}_{w,i}) = \begin{cases} 1 & \text{if there exists } \ell \text{ with } w = u_\ell \text{ and } f(u_\ell) = i \\ 0 & \text{otherwise,} \end{cases}$$

then we have $v_M(\varphi) = 1$ for all $\varphi \in \Gamma_0$. Thus, Γ_0 is satisfiable. By the Compactness Theorem, it follows that Γ is satisfiable.

Fix a truth assignment $M\colon P \to \{0,1\}$ such that $v_M(\varphi) = 1$ for all $\varphi \in \Gamma$. Notice that for each $u \in V$, there exists a unique i such that $M(\mathsf{A}_{u,i}) = 1$ because of the first two sets in the definition of Γ. If we define $f\colon V \to [k]$ by letting $f(u)$ be the unique i such that $v(\mathsf{A}_{u,i}) = 1$, then whenever $(u,w) \in E$, we have that $f(u) \neq f(w)$ (because of the third set in the definition of Γ). Therefore, G is k-colorable. □

Corollary 3.6.4. *Every (possibly infinite) planar graph is 4-colorable.*

Proof. Since every subgraph of a planar graph is planar, this follows trivially from the previous proposition and the highly nontrivial theorem that every finite planar graph is 4-colorable. □

Our next result is about infinite binary trees. We could code binary trees as connected acyclic graphs with certain degree restrictions, but for our purposes here, it will be more convenient to think about them differently. We start with a root, and at each node, we can have at most 2 children. It is then natural to code the two potential children of a node using two symbols, like 0 for left and 1 for right. In this way, we can uniquely find our place in a tree using a finite sequence of 0's and 1's. As a result, we might as well code trees by these binary sequences.

Definition 3.6.5. *A set $T \subseteq \{0,1\}^*$ is called a* tree *if whenever $\sigma \in T$ and $\tau \preceq \sigma$, we have $\tau \in T$.*

For example, the set $\{\lambda, 0, 1, 00, 01, 011, 0110, 0111\}$ is a tree.

Theorem 3.6.6 (Weak König's Lemma)**.** *Every infinite tree has an infinite branch. In other words, if $T \subseteq \{0,1\}^*$ is a tree with infinitely many elements, then there exists an $f\colon \mathbb{N} \to \{0,1\}$ such that $f \restriction [n] \in T$ for all $n \in \mathbb{N}$.*

Proof. For each $n \in \mathbb{N}$, let $T_n = \{\sigma \in T : |\sigma| = n\}$. Notice that each T_n is finite, because the set $\{0,1\}^n$ is finite. Since T is infinite, there must be infinitely many $n \in \mathbb{N}$ such that $T_n \neq \emptyset$. Since T is tree, and hence closed under initial segments, we know that if $T_n \neq \emptyset$, then $T_m \neq \emptyset$ for all $m < n$. Combining these facts, it follows that $T_n \neq \emptyset$ for all $n \in \mathbb{N}$.

Let $P = \{\mathsf{A}_\sigma : \sigma \in T\}$, and let Γ be the union of the following sets:

- $\left\{ \bigvee_{\sigma \in T_n} \mathsf{A}_\sigma : n \in \mathbb{N} \right\}$.
- $\{\neg(\mathsf{A}_\sigma \wedge \mathsf{A}_\tau) : \sigma, \tau \in T_n \text{ and } \sigma \neq \tau\}$.
- $\{\mathsf{A}_\sigma \to \mathsf{A}_\tau : \sigma, \tau \in T, \tau \preceq \sigma\}$.

We use the Compactness Theorem to show that Γ is satisfiable. Suppose that $\Gamma_0 \subseteq \Gamma$ is finite. Let $\Sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_k\}$ be all of the elements $\sigma \in \{0,1\}^*$ such that A_σ occurs in some element of Γ_0. Let $n =$

$\max\{|\sigma_1|, |\sigma_2|, \ldots, |\sigma_k|\}$. Since $T_n \neq \emptyset$, we may fix $\tau \in T_n$. If we define a truth assignment $M\colon P \to \{0,1\}$ by

$$M(\mathsf{A}_\sigma) = \begin{cases} 1 & \text{if } \sigma \preceq \tau \\ 0 & \text{otherwise,} \end{cases}$$

then we see that $v_M(\varphi) = 1$ for all $\varphi \in \Gamma_0$. Thus, Γ_0 is satisfiable. By the Compactness Theorem, it follows that Γ is satisfiable.

Fix a truth assignment $M\colon P \to \{0,1\}$ such that $v_M(\varphi) = 1$ for all $\varphi \in \Gamma$. Notice that for each $n \in \mathbb{N}^+$, there exists a unique $\sigma \in T_n$ such that $v(\mathsf{A}_\sigma) = 1$ because of the first two sets in the definition of Γ. For each n, denote the unique such σ by ρ_n and notice that $\rho_m \preceq \rho_n$ whenever $m \leq n$. Define $f\colon \mathbb{N} \to \{0,1\}$ by letting $f(n) = \rho_{n+1}(n)$. We then have that $f \restriction [n] = \rho_n \in T$ for all $n \in \mathbb{N}$. □

We end with an interesting algebraic application about abelian groups. Since we will only discuss abelian groups in this section, we will use $+$ for the binary operation, 0 for the identity, and $-a$ for the inverse of a. We begin with a definition that captures when we can put a linear (or total) ordering on the elements of the group that respects the binary operation.

Definition 3.6.7. *An* ordered abelian group *is an abelian group $(A, +, 0)$ together with a relation $\leq$ on A^2 with the following properties:*

1. *$\leq$ is a linear ordering on A, i.e. we have the following:*
   - *For all $a \in A$, we have $a \leq a$.*
   - *For all $a, b \in A$, either $a \leq b$ or $b \leq a$.*
   - *If $a \leq b$ and $b \leq a$, then $a = b$.*
   - *If $a \leq b$ and $b \leq c$, then $a \leq c$.*
2. *If $a \leq b$ and $c \in A$, then $a + c \leq b + c$.*

For example, $(\mathbb{Z}, +, 0)$ with its usual ordering is an ordered abelian group. Similarly, both $(\mathbb{Q}, +, 0)$ and $(\mathbb{R}, +, 0)$ are ordered abelian groups with their usual orderings. Recall that given two groups G and H, we can form the direct product $G \times H$, where the operation on $G \times H$ happens componentwise. In fact, we can form the direct product $G_1 \times G_2 \times \cdots \times G_n$ of finitely many groups (or even infinitely many). By taking the direct product of $\mathbb{Z}$ with itself a finite number n many times, we obtain an abelian group $\mathbb{Z}^n$. It turns out that we can equip $\mathbb{Z}^n$ with several interesting orderings, but we focus on one here. Define $\leq$ on $\mathbb{Z}^n$ by using the lexicographic, or dictionary, ordering. In other words, given elements $\vec{a} = (a_1, a_2, \ldots, a_n)$ and $\vec{b} = (b_1, b_2, \ldots, b_n)$ in $\mathbb{Z}^n$, say that $\vec{a} \leq \vec{b}$ if either of the following holds:

1. $\vec{a} = \vec{b}$, i.e. $a_i = b_i$ for all i.
2. $\vec{a} \neq \vec{b}$, and if i is least such that $a_i \neq b_i$, then $a_i <_{\mathbb{Z}} b_i$.

We can also state this by saying that $\vec{a} \leq \vec{b}$ if either $\vec{b} - \vec{a}$ is the zero vector, or the first nonzero element of $\vec{b} - \vec{a}$ is positive. With this ordering, it's straightforward to check that $(\mathbb{Z}^n, +, 0)$ is an ordered abelian group. In fact, it's relatively easy to generalize the construction to show that if $G_1, G_2, \ldots, G_n$ are all ordered abelian groups, then the direct product $G_1 \times G_2 \times \cdots \times G_n$ equipped with the lexicographic ordering is an ordered abelian group.

We want to understand what general ordered abelian groups look like, and which abelian groups we can equip with an ordering. To work toward this goal, we start with a simple property of ordered abelian groups.

Proposition 3.6.8. *Let $(A, +, 0, \leq)$ be an ordered abelian group. If $a \leq b$ and $c \leq d$, then $a + c \leq b + d$.*

Proof. Let $a, b, c, d \in A$ be arbitrary with $a \leq b$ and $c \leq d$. Since $a \leq b$ and $c \in A$, we know that $a+c \leq b+c$. Similarly, since $c \leq d$ and $b \in A$, we have $c+b \leq d+b$. Using the fact that $+$ is commutative, it follows that $b+c \leq b+d$. Finally, since we have both $a+c \leq b+c$ and also $b+c \leq b+d$, we can use the transitivity of $\leq$ to conclude that $a+c \leq b+d$. □

Now whenever we have a linear ordering $\leq$ on a set A, we can define a corresponding strict ordering $<$.

Proposition 3.6.9. *Suppose that $(A, +, 0, \leq)$ is an ordered abelian group. Define $<$ by letting $a < b$ if $a \leq b$ and $a \neq b$. We then have the following properties:*

1. *For all $a, b \in A$, exactly one of $a < b$, $a = b$, or $b < a$ holds.*
2. *If $a < b$ and $c \in A$, then $a + c < b + c$.*

Proof.

1. Let $a, b \in A$ be arbitrary. We first show that at least one of the three conditions holds. Assume then that $a \neq b$. By definition, we know that either $a \leq b$ or $b \leq a$ holds. If the former case we have $a < b$, while in the latter we have $b < a$.

   We now show that at most one holds. Clearly, we can't have both $a < b$ and $a = b$, nor can we have both $a = b$ and $b < a$. Suppose then that we have both $a < b$ and $b < a$. We would then have both $a \leq b$ and $b \leq a$, hence $a = b$, a contradiction.

2. Since $a < b$, we know that $a \leq b$, and hence $a + c \leq b + c$. Now if $a + c = b + c$, then by adding $-c$ to both sides we would have $a = b$, which is a contradiction. Therefore, $a + c < b + c$.

□

We are now ready to establish a simple restriction on the algebraic structure of any ordered abelian group.

Proposition 3.6.10. *In any ordered abelian group, every nonzero element has infinite order.*

Proof. Let $(A, +, 0, \leq)$ be an ordered abelian group. Let $a \in A$ be arbitrary with $a \neq 0$. For any $n \in \mathbb{N}^+$, let $n \cdot a$ be the result of adding a to itself n times in the abelian group. Now $0 \in A$ and $a \neq 0$, so we have two possible cases:

- *Case 1:* Suppose that $0 < a$. Adding a to both sides we conclude that $a < a + a$. Using the fact that $0 < a$ together with transitivity, it follows that $0 < a + a$. If we add a to both sides again and follow the same argument, we conclude that $0 < a + a + a$. From here, a simple induction establishes that $0 < n \cdot a$ for all $n \in \mathbb{N}^+$. In particular, $n \cdot a \neq 0$ for all $n \in \mathbb{N}^+$, so a has infinite order.
- *Case 2:* Suppose that $a < 0$. Following the logic in Case 1, a simple induction shows that $n \cdot a < 0$ for all $n \in \mathbb{N}^+$. In particular, $n \cdot a \neq 0$ for all $n \in \mathbb{N}^+$, so a has infinite order.

Therefore, every nonidentity element of A has infinite order. □

Somewhat surprisingly, the converse to this statement is also true, i.e. given an abelian group $(A, +, 0)$ in which every nonzero element has infinite order, we can find an ordering $\leq$ such that $(A, +, 0, \leq)$ is an ordered abelian group. Attacking this problem directly is difficult, as it's unclear how to define an ordering $\leq$ on a general group, using only the assumption that each nonidentity element has infinite order. The key idea is to use the Compactness Theorem to restrict to an appropriate "finite" case. We will need the following important algebraic result.

Theorem 3.6.11 (Fundamental Theorem of Finitely Generated Abelian Groups)**.** *Let G be a finitely generated abelian group. There exists $n \in \mathbb{N}$ and $m_1, m_2, \ldots, m_k \in \mathbb{N}^+$ with*

$$A \cong \mathbb{Z}^n \times \mathbb{Z}/m_1\mathbb{Z} \times \mathbb{Z}/m_2\mathbb{Z} \times \cdots \times \mathbb{Z}/m_k\mathbb{Z}.$$

In fact, it is possible (thought not necessary for our purposes) to say more. For example, one can choose $m_1, m_2, \ldots, m_k \in \mathbb{N}^+$ with $m_i \mid m_{i+1}$ for all i with $1 \leq i < k$. Alternatively, one can choose the m_i to be prime powers. Consult a standard algebra book for details.

Theorem 3.6.12. *If $(A, +, 0)$ is an abelian group in which every nonzero element has infinite order, we can find an ordering $\leq$ such that $(A, +, 0, \leq)$ is an ordered abelian group.*

Proof. We first prove the result for any finitely generated abelian group A. Given such a group A we know from the Fundamental Theorem of Finitely-Generated Abelian Groups that A must be isomorphic to $\mathbb{Z}^n$ for some $n \in \mathbb{N}^+$, because each $\mathbb{Z}/m\mathbb{Z}$ for $m \geq 2$ has nonidentity elements of finite order. Since $\mathbb{Z}^n$ can be equipped with the lexicographic ordering, we can transfer this ordering across the isomorphism to order A.

Suppose now that A is an arbitrary torsion-free abelian group. Let P be the set $\{\mathsf{L}_{a,b} : a, b \in A\}$ and let Γ be the union of the following sets:

- $\{\mathsf{L}_{a,a} : a \in A\}$.
- $\{\mathsf{L}_{a,b} \vee \mathsf{L}_{b,a} : a, b \in A\}$.
- $\{\neg(\mathsf{L}_{a,b} \wedge \mathsf{L}_{b,a}) : a, b \in A \text{ with } a \neq b\}$.
- $\{(\mathsf{L}_{a,b} \wedge \mathsf{L}_{b,c}) \to \mathsf{L}_{a,c} : a, b, c \in A\}$.
- $\{\mathsf{L}_{a,b} \to \mathsf{L}_{a+c,b+c} : a, b, c \in A\}$.

We show that Γ is satisfiable. By Compactness, it suffices to show that any finite subset of Γ is satisfiable. Suppose that $\Gamma_0 \subseteq \Gamma$ is finite, and let S be the finite subset of A consisting of all elements of A appearing as a subscript of a symbol occurring in Γ_0. Let B be the subgroup of A generated by S. We then have that B is a finitely generated torsion-free abelian group, so from above we may fix an ordering $\leq$ on it. If we define a truth assignment $v \colon P \to \{0, 1\}$ by

$$M(\mathsf{L}_{a,b}) = \begin{cases} 1 & \text{if } a \leq b \\ 0 & \text{otherwise.} \end{cases}$$

we see that $v_M(\varphi) = 1$ for all $\varphi \in \Gamma_0$. Thus, Γ_0 is satisfiable. By the Compactness Theorem, we conclude that Γ is satisfiable.

Fix a truth assignment $M \colon P \to \{0, 1\}$ such that $v_M(\gamma) = 1$ for all $\gamma \in \Gamma$. Define $\leq$ on A^2 by letting $a \leq b$ if and only if $M(\mathsf{L}_{a,b}) = 1$. We then have that $\leq$ is an ordering on A. □

In the parlance of group theory, a group in which every element has infinite order is called *torsion-free*. Thus, we can state the above results as stating that an abelian group can be ordered if and only if it is torsion-free.

3.7 Exercises

1. Definition 3.1.16 defines a function $Subform \colon Form_P \to \mathcal{P}(Form_P)$ recursively as follows:
   - $Subform(\mathsf{A}) = \{\mathsf{A}\}$ for all $\mathsf{A} \in P$.
   - $Subform(\neg\varphi) = \{\neg\varphi\} \cup Subform(\varphi)$.

- $Subform(\diamond\varphi\psi) = \{\diamond\varphi\psi\} \cup Subform(\varphi) \cup Subform(\psi)$ for each $\diamond \in \{\wedge, \vee, \rightarrow\}$.

Suppose that $\varphi, \psi \in Form_P$ and that ψ is a substring of φ (i.e. there exists $\theta, \rho \in Sym_P^*$ such that $\varphi = \theta\psi\rho$). Show that $\psi \in Subform(\varphi)$.

2. Given any $\theta, \gamma \in Form_P$, Definition 3.1.17 describes a function $Subst_\gamma^\theta \colon Form_P \to Form_P$ (intuitively substituting θ for all occurrences of γ), which is defined recursively as follows:

- $Subst_\gamma^\theta(\mathsf{A}) = \begin{cases} \theta & \text{if } \gamma = \mathsf{A} \\ \mathsf{A} & \text{otherwise.} \end{cases}$
- $Subst_\gamma^\theta(\neg\varphi) = \begin{cases} \theta & \text{if } \gamma = \neg\varphi \\ \neg Subst_\gamma^\theta(\varphi) & \text{otherwise.} \end{cases}$
- $Subst_\gamma^\theta(\diamond\varphi\psi) = \begin{cases} \theta & \text{if } \gamma = \diamond\varphi\psi \\ \diamond Subst_\gamma^\theta(\varphi) Subst_\gamma^\theta(\psi) & \text{otherwise.} \end{cases}$

  for each $\diamond \in \{\wedge, \vee, \rightarrow\}$.

Show that if $M \colon P \to \{0,1\}$ is a truth assignment with $v_M(\theta) = v_M(\gamma)$, then $v_M(\varphi) = v_M(Subst_\gamma^\theta(\varphi))$ for every $\varphi \in Form_P$.

3. Let $Form_P^- = G(Sym_P^*, P, \{h_\neg, h_\wedge, h_\vee\})$. Define a function $Dual \colon Form_P^- \to Form_P^-$ recursively as follows:

- $Dual(\mathsf{A}) = \neg\mathsf{A}$ for all $\mathsf{A} \in P$.
- $Dual(\neg\varphi) = \neg Dual(\varphi)$.
- $Dual(\wedge\varphi\psi) = \vee Dual(\varphi) Dual(\psi)$.
- $Dual(\vee\varphi\psi) = \wedge Dual(\varphi) Dual(\psi)$.

Show that $Dual(\varphi)$ is semantically equivalent to $\neg\varphi$ for all $\varphi \in Form_P^-$.

4. We can extend the notion of semantic equivalence to sets of formulas.

Definition 3.7.1. *Let $\Gamma_1, \Gamma_2 \subseteq Form_P$. We say that Γ_1 and Γ_2 are* semantically equivalent *if $\Gamma_1 \vDash \gamma_2$ for all $\gamma_2 \in \Gamma_2$ and $\Gamma_2 \vDash \gamma_1$ for all $\gamma_1 \in \Gamma_1$. Notice that this is equivalent to saying that whenever $M \colon P \to \{0,1\}$ is a truth assignment, then $v_M(\gamma_1) = 1$ for all $\gamma_1 \in \Gamma_1$ if and only if $v_M(\gamma_2) = 1$ for all $\gamma_2 \in \Gamma_2$.*

Definition 3.7.2. *Let $\Gamma \subseteq Form_P$. We say that Γ is* independent *if there is no $\varphi \in \Gamma$ such that $\Gamma \backslash \{\varphi\} \vDash \varphi$. Notice that this is equivalent to saying that Γ is not semantically equivalent to any proper subset.*

(a) Show that if Γ is finite, then Γ has an independent semantically equivalent subset.

(b) Show that there exists a set P and an infinite set $\Gamma \subseteq Form_P$ which has no independent semantically equivalent subset.

5. (**) Using the definition in the previous problem, show that every countable set is semantically equivalent to an independent set.

6. Let $P = \{\mathsf{A}_0, \mathsf{A}_1, \mathsf{A}_2, \mathsf{A}_3, \mathsf{A}_4, \mathsf{A}_5, \mathsf{A}_6\}$. Show that there exists a boolean function $f \colon \{0,1\}^7 \to \{0,1\}$ such that $Depth(\varphi) \geq 5$ for all φ with $B_\varphi = f$.

Hint: Do a counting argument.

7. For each $\varphi \in Form_P$, give a deduction showing that $\varphi \vdash \neg\neg\varphi$.

8. (a) For each $\varphi, \psi \in Form_P$, give a deduction showing that $\neg\varphi \vdash \neg(\varphi \wedge \psi)$.

   (b) For each $\varphi, \psi \in Form_P$, give a deduction showing that $\neg(\varphi \wedge \psi) \vdash (\neg\varphi) \vee (\neg\psi)$.

9. (a) In the proof of the Soundness Theorem (Theorem 3.5.2), we showed how statement (2) followed easily from statement (1). Show that this implication can be reversed. That is, give a short proof that "If $\Gamma \vdash \varphi$, then $\Gamma \vDash \varphi$" using the assumption "Every satisfiable set of formulas is consistent".

   (b) In the proof of the Completeness Theorem (Theorem 3.5.2), we showed how statement (2) followed easily from statement (1). Show that this implication can be reversed. That is, give a short proof that "Every consistent set of formulas is satisfiable" using the assumption "If $\Gamma \vDash \varphi$, then $\Gamma \vdash \varphi$".

   (c) Show how to obtain each version of the Compactness Theorem from the other using only short semantic arguments.

10. Suppose that $\theta \vdash \gamma$ and $\gamma \vdash \theta$. Show that if $\Gamma \vdash \varphi$, then $\Gamma \vdash Subst^{\theta}_{\gamma}(\varphi)$.

    Hint: You don't need to stay on the syntactic side. Don't be a hero.

11. (a) Suppose that we eliminate the $\rightarrow I$ rule, the $\neg PC$ rule, and the $Contr$ rule. Show that the Completeness Theorem no longer holds.

    (b) Suppose that we eliminate the $\neg PC$ rule and the $Contr$ rule. Show that the Completeness Theorem no longer holds.

    Hint: For part a, notice that these 3 rules are the only ones that let us completely remove a formula from the left-hand side. State precisely what this gives you, and how it establishes the result. Part (b) will require a bit more insight, but use the idea of part (a) as a guide.

12. Given two partial orderings $\leq$ and $\leq'$ on a set B, we say that $\leq'$ *extends* $\leq$ if whenever $a \leq b$, we have $a \leq' b$. In other words, the set of ordered pairs that satisfies the relation $\leq$ is a subset of the set of ordered pairs that satisfies $\leq'$. Recall that a linear ordering is a partial ordering with the extra properties that for all $a, b \in B$, either $a \leq b$ or $b \leq a$.

    (a) Show that if $(B, \leq)$ is a *finite* partial ordering (i.e. where B is finite), then there exists a linear ordering $\leq'$ of B extending $\leq$.

    (b) Show that if $(B, \leq)$ is any partial ordering, then there exists a linear ordering $\leq'$ of B extending $\leq$.

13. Let P be a partial ordering. A *chain* C in $P$is a subset of P such that for all $a, b \in C$, either $a \leq b$ or $b \leq a$. An *antichain* in P is a set $A \subseteq P$ such that no two distinct elements are comparable (i.e. for all $a, b \in A$, if $a \neq b$, then $a \not\leq b$ and $b \not\leq a$).

    (a) Suppose that $k \in \mathbb{N}^+$ and P is a (possibly infinite) partial ordering such that every finite subset of P is the union of k chains. Show that P is the union of k chains.

    (b) A theorem of finite combinatorics, known as Dilworth's Theorem, says that in a *finite* partial ordering P, the maximum size of an an antichain equals the minimum number of chains whose union equals P. Use this result and part a to show that if P is a (possibly infinite) partial ordering such that the maximum size of an an antichain equals $k \in \mathbb{N}^+$, then P is the union of k chains.

14. (**) The final two problems outline proofs of the Compactness Theorem that avoid using a notion of syntactic implication. The first one mimics the essential ideas in the proof of Completeness given in class, but stays semantic the whole way. In this problem, we say a set $\Gamma \subseteq Form_P$ is *finitely satisfiable* if every finite subset of Γ is satisfiable.

(a) Show that if $\Gamma \subseteq Form_P$ is finitely satisfiable and $\varphi \in Form_P$, then either $\Gamma \cup \{\varphi\}$ is finitely satisfiable or $\Gamma \cup \{\neg\varphi\}$ is finitely satisfiable.

(b) Show that if $\Gamma \subseteq Form_P$ is finitely satisfiable, then there exists $\Delta \supseteq \Gamma$ which is both complete and finitely satisfiable.

(c) Show that if Δ is both complete and finitely satisfiable, then Δ is satisfiable.

(d) Establish the Compactness Theorem.

15. (**) This proof of the Compactness Theorem uses ideas from topology. For any set P, let T_P be the topological space $\{0,1\}^P$ (which can be viewed either as the product of copies of $\{0,1\}$ indexed by P, or as truth assignments on P), where we give $\{0,1\}$ the discrete topology and $\{0,1\}^P$ the corresponding product topology.

(a) Viewing elements of T_P as truth assignments, show that the set $\{M \in T_P : v_M(\varphi) = 1\}$ is closed for each $\varphi \in Form_P$.

(b) Use Tychonov's Theorem to prove the Compactness Theorem.

(c) Without applying Tychonov's Theorem, use the Compactness Theorem to prove that T_P is compact for every P.

Chapter 4

First-Order Logic: Languages and Structures

Now that we have successfully worked through several important aspects of propositional logic, it is time to move on to a much more substantial and important logic: first-order logic. As summarized in the introduction, the general idea is as follows. Many areas of mathematics deal with mathematical structures consisting of special constants, relations, and functions, together with certain axioms that these objects obey. We want our logic to be able to handle many different types of situations, so we allow ourselves to vary the number and types of the symbols. For example, in group theory, we have a special identity element and a binary function corresponding to the group operation. If we wanted, we could also add in a unary function corresponding to the inverse operation. For ring theory, we have two constants for 0 and 1 along with two binary operations for addition and multiplication (and possibly a unary function for additive inverses). For partial orderings, we have one binary relation. Any such choice gives rise to a *language*.

Once we've fixed such a language, we can build up formulas that will express something meaningful. As an example, we mentioned in the introduction that

$$\forall \mathsf{x} \forall \mathsf{y} (\mathsf{f}(\mathsf{x}, \mathsf{y}) = \mathsf{f}(\mathsf{y}, \mathsf{x}))$$

is a formula in the language of group theory. Now in isolation, this formula is neither true nor false, just like the propositional formula $\mathsf{A} \wedge (\mathsf{B} \vee \mathsf{C})$ is neither true nor false without a truth assignment. The analogue to a truth assignment in first-order logic is called a *structure*. In this setting, a structure provides an interpretation for all of the symbols, and once we fix a structure (i.e. once we fix an actual group, ring, partial ordering, etc.), we can ask whether the formula is true in that world.

Building up the fundamental definitions (like formulas and structures) will take some time. Our experience with propositional logic will certainly help here, but the complexity is considerably higher.

4.1 Terms and Formulas

Since our logic will have quantifiers, the first thing that we need is a collection of variables, like the x and y in the formula $\forall \mathsf{x} \forall \mathsf{y} (\mathsf{f}(\mathsf{x}, \mathsf{y}) = \mathsf{f}(\mathsf{y}, \mathsf{x}))$. Since our formulas will consist of only finitely many characters, and since we will want to ensure that we always have an extra variable around if we need it, we start with the fixing a large enough set.

Definition 4.1.1. *Fix a countably infinite set Var called* variables.

We now define a language. As mentioned, we want to allow flexibility in the number and types of symbols here, depending on what field of mathematics we want to model.

Definition 4.1.2. *A* first-order language*, or simply a* language*, consists of the following:*

1. *A set $\mathcal{C}$ of* constant symbols*.*
2. *A set $\mathcal{F}$ of* function symbols *together with a function $Arity_{\mathcal{F}} \colon \mathcal{F} \to \mathbb{N}^+$.*
3. *A set $\mathcal{R}$ of* relation symbols *together with a function $Arity_{\mathcal{R}} \colon \mathcal{R} \to \mathbb{N}^+$.*

We also assume that $\mathcal{C}$, $\mathcal{R}$, $\mathcal{F}$, Var, and $\{\forall, \exists, =, \neg, \wedge, \vee, \to\}$ are pairwise disjoint. For each $k \in \mathbb{N}^+$, we let

$$\mathcal{F}_k = \{\mathsf{f} \in \mathcal{F} : Arity_{\mathcal{F}}(\mathsf{f}) = k\},$$

and we let

$$\mathcal{R}_k = \{\mathsf{R} \in \mathcal{R} : Arity_{\mathcal{R}}(\mathsf{R}) = k\}.$$

Finally, given a language $\mathcal{L}$, we let $Sym_{\mathcal{L}} = \mathcal{C} \cup \mathcal{R} \cup \mathcal{F} \cup Var \cup \{\forall, \exists, =, \neg, \wedge, \vee, \to\}$.

Let's consider some examples of languages. Suppose that we want to create a language that is suitable for group theory. One option would be to let $\mathcal{C} = \{\mathsf{e}\}$, let $\mathcal{F} = \{\mathsf{f}\}$ where f has arity 2, and we let $\mathcal{R} = \emptyset$. Intuitively, the symbol e will represent the identity and f will represent the group operation. Alternatively, we could let $\mathcal{C} = \{\mathsf{e}\}$, let $\mathcal{F} = \{\mathsf{f}, \mathsf{g}\}$ where f has arity 2 and g has arity 1, and let $\mathcal{R} = \emptyset$. In this setting, g will represent the inverse operation in the group. In other words, even when we have an area of mathematics in mind, there may be several suitable languages that we will have to choose between. At this point, it may seem like a matter of taste. However, we will eventually see how the choice of language affects other concepts that we will define (see Section 4.3).

From here on, we will call the first language described above (the one with only 1 function symbol) the *restricted group theory* language, and we will call the latter simply the *group theory language*. Note that the particular names for symbols do not matter much. In other words, we can use a binary relation symbol h instead of f in the restricted group theory language. Alternatively, we can use a more natural binary relation symbol like $\cdot$. Thus, we might write "consider the restricted group theory language $\{\mathsf{e}, \cdot\}$".

Now consider the language where $\mathcal{C} = \emptyset$, $\mathcal{F} = \emptyset$, and $\mathcal{R} = \{\mathsf{R}\}$ where R has arity 2. In other words, we have just one binary relation symbol. As mentioned in the introduction, this language is suitable for the theory of partial orderings. Notice that it would also serve as a language for the theory of equivalence relations. Thus, one language can be repurposed for different areas of mathematics.

Once we have chosen a language, we have fixed the collection of symbols that are available. The first major task is to determine how to generate formulas, like $\forall \mathsf{x} \forall \mathsf{y}(\mathsf{f}(\mathsf{x}, \mathsf{y}) = \mathsf{f}(\mathsf{y}, \mathsf{x}))$ in the group theory language. Before doing this, however, we need a way to name elements. Intuitively, our constant symbols and variables name elements once we've fixed an interpretation (i.e. once we've fixed a *structure*, which we will define in the next section). However, we can form more complex names for elements if we have function symbols. For example, in our group theory language, if x and y are variables, then the $\mathsf{f}(\mathsf{x}, \mathsf{y})$ in the above formula would also name an element. We can then go on to form more complex names from here, such as $\mathsf{f}(\mathsf{f}(\mathsf{x}, \mathsf{y}), \mathsf{x})$ or $\mathsf{f}(\mathsf{e}, \mathsf{g}(\mathsf{y}))$. The idea then is to start with the constant symbols and variables, and then generate new names by repeatedly applying function symbols. As we've started to appreciate from our exposure to Polish notation, it turns out that we can avoid the parentheses and commas. Putting it all together, we obtain the following definition.

Definition 4.1.3. *Let $\mathcal{L}$ be a language. For each $\mathsf{f} \in \mathcal{F}_k$, define $h_{\mathsf{f}} \colon (Sym_{\mathcal{L}}^*)^k \to Sym_{\mathcal{L}}^*$ by letting*

$$h_{\mathsf{f}}(\sigma_1, \sigma_2, \ldots, \sigma_k) = \mathsf{f}\sigma_1\sigma_2 \cdots \sigma_k.$$

We then define

$$Term_{\mathcal{L}} = G(Sym_{\mathcal{L}}^*, \mathcal{C} \cup Var, \{h_{\mathsf{f}} : \mathsf{f} \in \mathcal{F}\}).$$

and call the elements of $Term_{\mathcal{L}}$ the terms *of the language.*

Now that we have terms, which intuitively name elements once we've fixed an interpretation, we can start to think about formulas. In propositional logic, our most basic formulas were the symbols from P themselves. In this new setting, the basic formulas are more interesting. The idea is that the most fundamental things that we can say are whether or not two elements are equal, and whether or not a k-tuple is in the interpretation of some relation symbol $\mathsf{R} \in \mathcal{R}_k$.

Definition 4.1.4. *Let $\mathcal{L}$ be a language. We let*

$$AtomicForm_{\mathcal{L}} = \{\mathsf{R}t_1t_2\cdots t_k : k \in \mathbb{N}^+, \mathsf{R} \in \mathcal{R}_k, \text{ and } t_1, t_2, \ldots, t_k \in Term_{\mathcal{L}}\} \cup \{= t_1t_2 : t_1, t_2 \in Term_{\mathcal{L}}\},$$

and call the elements of $AtomicForm_{\mathcal{L}}$ the atomic formulas *of the language.*

Starting with atomic formulas, we now generate more complex formulas by introducing our old propositional logic connectives, and by allowing the use of quantifiers.

Definition 4.1.5. *Let $\mathcal{L}$ be a language. Define a unary function $h_\neg$ and binary functions $h_\wedge, h_\vee$, and $h_\to$ on $Sym^*_{\mathcal{L}}$ as follows:*

$$\begin{aligned} h_\neg(\sigma) &= \neg\sigma \\ h_\wedge(\sigma, \tau) &= \wedge\sigma\tau \\ h_\vee(\sigma, \tau) &= \vee\sigma\tau \\ h_\to(\sigma, \tau) &= \to\sigma\tau. \end{aligned}$$

Also, for each $\mathsf{x} \in Var$, define two unary functions $h_{\forall,\mathsf{x}}$ and $h_{\exists,\mathsf{x}}$ on $Sym^*_{\mathcal{L}}$ as follows:

$$\begin{aligned} h_{\forall,\mathsf{x}}(\sigma) &= \forall\mathsf{x}\sigma \\ h_{\exists,\mathsf{x}}(\sigma) &= \exists\mathsf{x}\sigma. \end{aligned}$$

Let

$$Form_{\mathcal{L}} = G(Sym^*_{\mathcal{L}}, AtomicForm_{\mathcal{L}}, \{h_\neg, h_\wedge, h_\vee, h_\to\} \cup \{h_{\forall,\mathsf{x}}, h_{\exists,\mathsf{x}} : \mathsf{x} \in Var\}).$$

As with propositional logic, we'd like to be able to define things recursively, so we need to check that our generating systems are free. Notice that in the construction of formulas, we have two generating systems around. We first generate all terms. With terms taken care of, we next describe the atomic formulas, and from them we generate all formulas. Thus, we'll need to prove that two generating systems are free. The general idea is to make use of the insights gained by proving the corresponding result for Polish notation in propositional logic.

Definition 4.1.6. *Let $\mathcal{L}$ be a language. Define $W\colon Sym^*_{\mathcal{L}} \to \mathbb{Z}$ as follows. We first define $w\colon Sym_{\mathcal{L}} \to \mathbb{Z}$ as follows:*

$$\begin{aligned} w(\mathsf{c}) &= 1 && \text{for all } \mathsf{c} \in \mathcal{C} \\ w(\mathsf{f}) &= 1 - k && \text{for all } \mathsf{f} \in \mathcal{F}_k \\ w(\mathsf{R}) &= 1 - k && \text{for all } \mathsf{R} \in \mathcal{R}_k \\ w(\mathsf{x}) &= 1 && \text{for all } \mathsf{x} \in Var \\ w(=) &= -1 \\ w(\mathsf{Q}) &= -1 && \text{for all } \mathsf{Q} \in \{\forall, \exists\} \\ w(\neg) &= 0 \\ w(\diamond) &= -1 && \text{for all } \diamond \in \{\wedge, \vee, \to\}. \end{aligned}$$

We then define W on all of $Sym^*_{\mathcal{L}}$ by letting $W(\lambda) = 0$ and letting $W(\sigma) = \sum_{i<|\sigma|} w(\sigma(i))$ for all $\sigma \in Sym^*_{\mathcal{L}} \backslash \{\lambda\}$.

As usual, notice that if $\sigma, \tau \in Sym^*_{\mathcal{L}}$, then $W(\sigma\tau) = W(\sigma) + W(\tau)$.

Proposition 4.1.7. *Let $\mathcal{L}$ be a language. For all $t \in Term_{\mathcal{L}}$, we have $W(t) = 1$.*

Proof. The proof is by induction on t. Notice first that $W(\mathsf{c}) = 1$ for all $\mathsf{c} \in \mathcal{C}$ and $W(\mathsf{x}) = 1$ for all $\mathsf{x} \in Var$. Suppose that $k \in \mathbb{N}^+$, $\mathsf{f} \in \mathcal{F}_k$, and $t_1, t_2, \ldots, t_k \in Term_{\mathcal{L}}$ are such that $W(t_i) = 1$ for all i. We then have that

$$\begin{aligned} W(\mathsf{f}t_1t_2\cdots t_k) &= W(\mathsf{f}) + W(t_1) + W(t_2) + \cdots + W(t_k) \\ &= (1-k) + 1 + 1 + \cdots + 1 && \text{(by induction)} \\ &= 1. \end{aligned}$$

The result follows by induction. □

Proposition 4.1.8. *If $t \in Term_{\mathcal{L}}$ and $\sigma \prec t$, then $W(\sigma) \leq 0$.*

Proof. The proof is by induction on t. For every $\mathsf{c} \in \mathcal{C}$, this is trivial because the only $\sigma \prec \mathsf{c}$ is $\sigma = \lambda$ and we have $W(\lambda) = 0$. Similarly, for every $\mathsf{x} \in Var$, the only $\sigma \prec \mathsf{x}$ is $\sigma = \lambda$ and we have $W(\lambda) = 0$.

Suppose that $k \in \mathbb{N}^+$, $\mathsf{f} \in \mathcal{F}_k$, and $t_1, t_2, \ldots, t_k \in Term_{\mathcal{L}}$ are such that the the statement is true for each t_i. We prove the result for $\mathsf{f}t_1t_2\cdots t_k$. Suppose that $\sigma \prec \mathsf{f}t_1t_2\cdots t_k$. If $\sigma = \lambda$, then $W(\sigma) = 0$. Otherwise, there exists $i < k$ and $\tau \prec t_i$ such that $\sigma = \mathsf{f}t_1t_2\cdots t_{i-1}\tau$, in which case

$$\begin{aligned} W(\sigma) &= W(\mathsf{f}) + W(t_1) + W(t_2) + \cdots + W(t_{i-1}) + W(\tau) \\ &= (1-k) + 1 + 1 + \cdots + 1 + W(\tau) && \text{(by Proposition 4.1.7)} \\ &= (1-k) + i + W(\tau) \\ &\leq (1-k) + i + 0 && \text{(by induction)} \\ &= 1 + (i-k) \\ &\leq 0. && \text{(since } i < k\text{)} \end{aligned}$$

Thus, the statement is true for $\mathsf{f}t_1t_2\cdots t_k$. □

Corollary 4.1.9. *If $t, u \in Term_{\mathcal{L}}$, then $t \not\prec u$.*

Proof. This follows by combining Proposition 4.1.7 and Proposition 4.1.8. □

Theorem 4.1.10. *The generating system $(Sym^*_{\mathcal{L}}, \mathcal{C} \cup Var, \{h_{\mathsf{f}} : \mathsf{f} \in \mathcal{F}\})$ is free.*

Proof. First notice that for all $\mathsf{f} \in \mathcal{F}$, we have that $\text{range}(h_{\mathsf{f}} \restriction (Term_{\mathcal{L}})^k) \cap (\mathcal{C} \cup Var) = \emptyset$ because all elements of $\text{range}(h_{\mathsf{f}})$ begin with f and we know that $\mathsf{f} \notin \mathcal{C} \cup Var$.

Let $\mathsf{f} \in \mathcal{F}_k$. Suppose that $t_1, t_2, \ldots, t_k, u_1, u_2, \ldots, u_k \in Term_{\mathcal{L}}$ and $h_{\mathsf{f}}(t_1, t_2, \ldots, t_k) = h_{\mathsf{f}}(u_1, u_2, \ldots, u_k)$. We then have $\mathsf{f}t_1t_2\cdots t_k = \mathsf{f}u_1u_2\cdots u_k$, hence $t_1t_2\cdots t_k = u_1u_2\cdots u_k$. Since $t_1 \prec u_1$ and $u_1 \prec t_1$ are both impossible by Corollary 4.1.9, it follows that $t_1 = u_1$. Thus, $t_2\cdots t_k = u_2\cdots u_k$, and so $t_2 = u_2$ for the same reason. Continuing in this fashion, we conclude that $t_i = u_i$ for all i. It follows that $h_{\mathsf{f}} \restriction (Term_{\mathcal{L}})^k$ is injective.

Finally notice that for any $\mathsf{f} \in \mathcal{F}_k$ and any $\mathsf{g} \in \mathcal{F}_\ell$ with $\mathsf{f} \neq \mathsf{g}$, we have that $\text{range}(h_{\mathsf{f}} \restriction (Term_{\mathcal{L}})^k) \cap \text{range}(h_{\mathsf{g}} \restriction (Term_{\mathcal{L}})^\ell) = \emptyset$ because all elements of $\text{range}(h_{\mathsf{f}} \restriction (Term_{\mathcal{L}})^k)$ begin with f while all elements of $\text{range}(h_{\mathsf{g}} \restriction (Term_{\mathcal{L}})^\ell)$ begin with g. □

Proposition 4.1.11. *If $\varphi \in Form_{\mathcal{L}}$, then $W(\varphi) = 1$.*

Proof. The proof is by induction on φ. We first show that $W(\varphi) = 1$ for all $\varphi \in AtomicForm_{\mathcal{L}}$. Suppose that φ is $\mathsf{R}t_1t_2\cdots t_k$ where $\mathsf{R} \in \mathcal{R}_k$ and $t_1, t_2, \ldots, t_k \in Term_{\mathcal{L}}$. We then have

$$\begin{aligned} W(\mathsf{R}t_1t_2\cdots t_k) &= W(\mathsf{R}) + W(t_1) + W(t_2) + \cdots + W(t_k) \\ &= (1-k) + 1 + 1 + \cdots + 1 && \text{(by Proposition 4.1.7)} \\ &= 1. \end{aligned}$$

Suppose that φ is $= t_1t_2$ where $t_1, t_2 \in Term_{\mathcal{L}}$. We then have

$$\begin{aligned} W(= t_1t_2) &= W(=) + W(t_1) + W(t_2) \\ &= -1 + 1 + 1 && \text{(by Proposition 4.1.7)} \\ &= 1. \end{aligned}$$

Thus, $W(\varphi) = 1$ for all $\varphi \in AtomicForm_{\mathcal{L}}$.

Suppose that $\varphi \in Form_{\mathcal{L}}$ is such that $W(\varphi) = 1$. We then have that

$$\begin{aligned} W(\neg\varphi) &= W(\neg) + W(\varphi) \\ &= 0 + 1 \\ &= 1. \end{aligned}$$

For any $\mathsf{Q} \in \{\forall, \exists\}$ and any $\mathsf{x} \in Var$ we also have

$$\begin{aligned} W(\mathsf{Q}\mathsf{x}\varphi) &= W(\mathsf{Q}) + W(\mathsf{x}) + W(\varphi) \\ &= -1 + 1 + 1 \\ &= 1. \end{aligned}$$

Suppose now that $\varphi, \psi \in Form_{\mathcal{L}}$ are such that $W(\varphi) = 1 = W(\psi)$, and $\diamond \in \{\wedge, \vee, \rightarrow\}$. We then have that

$$\begin{aligned} W(\diamond\varphi\psi) &= -1 + W(\varphi) + W(\psi) \\ &= -1 + 1 + 1 \\ &= 1. \end{aligned}$$

The result follows by induction. □

Proposition 4.1.12. *If $\varphi \in Form_{\mathcal{L}}$ and $\sigma \prec \varphi$, then $W(\sigma) \leq 0$.*

Proof. The proof is by induction on φ. We first show that the statement is true for all $\varphi \in AtomicForm_{\mathcal{L}}$. Suppose that φ is $\mathsf{R}t_1t_2\cdots t_k$ where $\mathsf{R} \in \mathcal{R}_k$ and $t_1, t_2, \ldots, t_k \in Term_{\mathcal{L}}$. Suppose that $\sigma \prec \mathsf{R}t_1t_2\cdots t_k$. If $\sigma = \lambda$, then $W(\sigma) = 0$. Otherwise, there exists $i < k$ and $\tau \prec t_i$ such that σ is $\mathsf{R}t_1t_2\cdots t_{i-1}\tau$, in which case

$$\begin{aligned} W(\sigma) &= W(\mathsf{R}) + W(t_1) + W(t_2) + \cdots + W(t_{i-1}) + W(\tau) \\ &= (1-k) + 1 + 1 + \cdots + 1 + W(\tau) && \text{(by Proposition 4.1.7)} \\ &= (1-k) + i + W(\tau) \\ &\leq (1-k) + i + 0 && \text{(by induction)} \\ &= 1 + (i-k) \\ &\leq 0. && \text{(since } i < k\text{)} \end{aligned}$$

Thus, the statement is true for $\mathsf{R}t_1t_2\cdots t_k$. The same argument works for $= t_1t_2$ where $t_1, t_2 \in Term_{\mathcal{L}}$, so the statement is true for all $\varphi \in AtomicForm_{\mathcal{L}}$.

Suppose that the statement is true for $\varphi \in Form_{\mathcal{L}}$. Suppose that $\sigma \prec \neg\varphi$. If $\sigma = \lambda$, then $W(\sigma) = 0$. Otherwise, $\sigma = \neg\tau$ for some $\tau \prec \varphi$, in which case

$$\begin{aligned} W(\sigma) &= W(\neg) + W(\tau) \\ &= 0 + W(\tau) \\ &\leq 0. && \text{(by induction)} \end{aligned}$$

Suppose now that $\mathsf{Q} \in \{\forall, \exists\}$, that $\mathsf{x} \in Var$, and that $\sigma \prec \mathsf{Q}\mathsf{x}\varphi$. If $\sigma = \lambda$, then $W(\sigma) = 0$, and if $\sigma = Q$, then $W(\sigma) = -1$. Otherwise, $\sigma = \mathsf{Q}\mathsf{x}\tau$ for some $\tau \prec \varphi$, in which case

$$\begin{aligned} W(\sigma) &= W(\mathsf{Q}) + W(\mathsf{x}) + W(\tau) \\ &= -1 + 1 + W(\tau) \\ &= 0 && \text{(by induction)} \end{aligned}$$

Suppose now that the the statement is true for $\varphi, \psi \in Form_{\mathcal{L}}$, and $\diamond \in \{\wedge, \vee, \rightarrow\}$. Suppose that $\sigma \prec \diamond\varphi\psi$. If $\sigma = \lambda$, then $W(\sigma) = 0$. If σ is $\diamond\tau$ for some $\tau \prec \varphi$, then

$$\begin{aligned} W(\sigma) &= W(\diamond) + W(\tau) \\ &= -1 + W(\tau) \\ &\leq -1. && \text{(by induction)} \end{aligned}$$

Otherwise, σ is $\diamond\varphi\tau$ for some $\tau \prec \psi$, in which case

$$\begin{aligned} W(\sigma) &= W(\diamond) + W(\varphi) + W(\tau) \\ &= -1 + 0 + W(\tau) && \text{(by Proposition 4.1.11)} \\ &\leq -1. && \text{(by induction)} \end{aligned}$$

Thus, the statement is true for $\diamond\varphi\psi$. □

Corollary 4.1.13. *If $\varphi, \psi \in Form_{\mathcal{L}}$, then $\varphi \not\prec \psi$.*

Proof. This follows by combining Proposition 4.1.11 and Proposition 4.1.12. □

Theorem 4.1.14. *The generating system $(Sym^*_{\mathcal{L}}, AtomicForm_{\mathcal{L}}, \{h_\neg, h_\wedge, h_\vee, h_\rightarrow\} \cup \{h_{\forall,\mathsf{x}}, h_{\exists,\mathsf{x}} : \mathsf{x} \in V\})$ is free.*

Proof. Similar to the others. □

Although we have a formal syntax using Polish notation, we will often write formulas in a more natural and intuitive way that employs parentheses and simple shortcuts in the interest of human readability. For example, we will typically write $\forall\mathsf{x}\forall\mathsf{y}(\mathsf{f}(\mathsf{x},\mathsf{y}) = \mathsf{f}(\mathsf{y},\mathsf{x}))$ instead of $\forall\mathsf{x}\forall\mathsf{y} = \mathsf{fxyfyx}$. We will also sometimes employ infix notation for functions. For example, if we view the restricted group theory language as having one constant symbol e and one binary function symbol $\cdot$, then we can informally write $\cdot$ between two arguments. In other words, instead of writing the proper formula $\forall\mathsf{x}\forall\mathsf{y} = \cdot\,\mathsf{xy}\cdot\mathsf{yx}$ in this language, we might refer to it by writing $\forall\mathsf{x}\forall\mathsf{y}(\mathsf{x}\cdot\mathsf{y} = \mathsf{y}\cdot\mathsf{x})$.

With these freeness results, we are now able to define functions recursively on $Term_{\mathcal{L}}$ and $Form_{\mathcal{L}}$. Since we use terms in our definition of atomic formulas, which are the basic formulas, we will often need to make two recursive definitions (on terms first, then on formulas) in order to define a function on formulas. Here's an example of how to define a function that produces the set of variables that occur in a given formula.

Definition 4.1.15. *Let $\mathcal{L}$ be a language.*

1. *We first define a function* $OccurVar\colon Term_{\mathcal{L}} \to \mathcal{P}(Var)$ *recursively as follows:*
   - $OccurVar(\mathsf{c}) = \emptyset$ *for all* $\mathsf{c} \in \mathcal{C}$.
   - $OccurVar(\mathsf{x}) = \{\mathsf{x}\}$ *for all* $\mathsf{x} \in Var$.
   - $OccurVar(\mathsf{f}t_1t_2\cdots t_k) = OccurVar(t_1) \cup OccurVar(t_2) \cup \cdots \cup OccurVar(t_k)$ *for all* $\mathsf{f} \in \mathcal{F}_k$ *and* $t_1, t_2, \ldots, t_k \in Term_{\mathcal{L}}$.
2. *We then define a function* $OccurVar\colon Form_{\mathcal{L}} \to \mathcal{P}(Var)$ *recursively as follows:*
   - $OccurVar(\mathsf{R}t_1t_2\cdots t_k) = OccurVar(t_1) \cup OccurVar(t_2) \cup \cdots \cup OccurVar(t_k)$ *for all* $\mathsf{R} \in \mathcal{R}_k$ *and* $t_1, t_2, \ldots, t_k \in Term_{\mathcal{L}}$.
   - $OccurVar(= t_1t_2) = OccurVar(t_1) \cup OccurVar(t_2)$ *for all* $t_1, t_2 \in Term_{\mathcal{L}}$.
   - $OccurVar(\neg\varphi) = OccurVar(\varphi)$ *for all* $\varphi \in Form_{\mathcal{L}}$.
   - $OccurVar(\diamond\varphi\psi) = OccurVar(\varphi) \cup OccurVar(\psi)$ *for each* $\diamond \in \{\wedge, \vee, \to\}$ *and* $\varphi, \psi \in Form_{\mathcal{L}}$.
   - $OccurVar(\mathsf{Q}\mathsf{x}\varphi) = OccurVar(\varphi) \cup \{\mathsf{x}\}$ *for each* $\mathsf{Q} \in \{\forall, \exists\}$, $\mathsf{x} \in Var$, *and* $\varphi \in Form_{\mathcal{L}}$.

Technically, we should probably use two different names for the functions above, since they have very different domains. However, there is little risk of confusion, so we just overload the function name. Although $OccurVar$ does produce the set of variables in a formula, notice that variables can occur in different ways within a formula. For example, if we are working in the group theory language, and we let φ be the formula $\forall\mathsf{y}(\mathsf{f}(\mathsf{x},\mathsf{y}) = \mathsf{f}(\mathsf{y},\mathsf{x}))$, then $OccurVar(\varphi) = \{\mathsf{x},\mathsf{y}\}$. However, notice that the x and y "sit" differently within the formula. Intuitively, the occurrences of y are *bound* by the quantifier, while the occurrences of x are *free* (i.e. not bound). We will have a great deal more to say about the distinction between free and bound variables, but we first define the recursive functions that produce the set of free and bound variables.

Definition 4.1.16. *Let* $\mathcal{L}$ *be a language.*

1. *We define a function* $FreeVar\colon Form_{\mathcal{L}} \to \mathcal{P}(Var)$ *recursively as follows.*
   - $FreeVar(\mathsf{R}t_1t_2\cdots t_k) = OccurVar(t_1) \cup OccurVar(t_2) \cup \cdots \cup OccurVar(t_k)$ *for all* $\mathsf{R} \in \mathcal{R}_k$ *and* $t_1, t_2, \ldots, t_k \in Term_{\mathcal{L}}$.
   - $FreeVar(= t_1t_2) = OccurVar(t_1) \cup OccurVar(t_2)$ *for all* $t_1, t_2 \in Term_{\mathcal{L}}$.
   - $FreeVar(\neg\varphi) = FreeVar(\varphi)$ *for all* $\varphi \in Form_{\mathcal{L}}$.
   - $FreeVar(\diamond\varphi\psi) = FreeVar(\varphi) \cup FreeVar(\psi)$ *for each* $\diamond \in \{\wedge, \vee, \to\}$ *and* $\varphi, \psi \in Form_{\mathcal{L}}$.
   - $FreeVar(\mathsf{Q}\mathsf{x}\varphi) = FreeVar(\varphi)\backslash\{\mathsf{x}\}$ *for each* $\mathsf{Q} \in \{\forall, \exists\}$, $\mathsf{x} \in Var$, *and* $\varphi \in Form_{\mathcal{L}}$.
2. *We define a function* $BoundVar\colon Form_{\mathcal{L}} \to \mathcal{P}(Var)$ *recursively as follows.*
   - $BoundVar(\mathsf{R}t_1t_2\cdots t_k) = \emptyset$ *for all* $\mathsf{R} \in \mathcal{R}_k$ *and* $t_1, t_2, \ldots, t_k \in Term_{\mathcal{L}}$.
   - $BoundVar(= t_1t_2) = \emptyset$ *for all* $t_1, t_2 \in Term_{\mathcal{L}}$.
   - $BoundVar(\neg\varphi) = BoundVar(\varphi)$ *for all* $\varphi \in Form_{\mathcal{L}}$.
   - $BoundVar(\diamond\varphi\psi) = BoundVar(\varphi) \cup BoundVar(\psi)$ *for each* $\diamond \in \{\wedge, \vee, \to\}$ *and* $\varphi, \psi \in Form_{\mathcal{L}}$.
   - $BoundVar(\mathsf{Q}\mathsf{x}\varphi) = BoundVar(\varphi) \cup \{\mathsf{x}\}$ *for each* $\mathsf{Q} \in \{\forall, \exists\}$, $\mathsf{x} \in Var$, *and* $\varphi \in Form_{\mathcal{L}}$.

Returning to our example formula φ in the group theory language equal to $\forall\mathsf{y}(\mathsf{f}(\mathsf{x},\mathsf{y}) = \mathsf{f}(\mathsf{y},\mathsf{x}))$, we now have that $FreeVar(\varphi) = \{\mathsf{x}\}$ and $BoundVar(\varphi) = \{\mathsf{y}\}$. However, notice that it is possible for a variable to be both free and bound. For example, if we work in the same language, but consider the formula ψ equal to $(\mathsf{f}(\mathsf{x},\mathsf{x}) = \mathsf{x}) \wedge \exists\mathsf{x}(\mathsf{x} = \mathsf{e})$, then we have $FreeVar(\psi) = \{\mathsf{x}\} = BoundVar(\psi)$. In other words, some occurrences of x are free and others are bound.

Definition 4.1.17. *Let $\mathcal{L}$ be a language and let $\varphi \in Form_{\mathcal{L}}$. We say that φ is an* $\mathcal{L}$-sentence*, or simply a* sentence*, if $FreeVar(\varphi) = \emptyset$. We let $Sent_{\mathcal{L}}$ be the set of sentences.*

As we will see, sentences will play an important role for us, since they do not have any "hanging" variables that are not captured by quantifiers.

4.2 Structures

Up until this point, all that we've dealt with in first-order logic are sequences of symbols without meaning. Sure, our motivation was to capture meaningful situations with our languages and the way we've described formulas, but all we've done so far is describe the grammar. If we want our formulas to actually express something, we need to set up a context in which to interpret them. In propositional logic, we needed truth assignments on P to give a "meaning" to a formula. Since we have quantifiers now, the first thing we'll need is a nonempty set M to serve as the domain of objects that the quantifiers range over. Once we've fixed that, we need to interpret the symbols of our language as actual elements of our set (in the case of constant symbols c), actual k-ary relations on M (in the case of k-ary relation symbols R), and actual k-ary functions on M (in the case of k-ary function symbols f).

Definition 4.2.1. *Let $\mathcal{L}$ be a language. An* $\mathcal{L}$-structure*, or simply a* structure*, is a sequence $\mathcal{M} = (M, g_{\mathcal{C}}, g_{\mathcal{F}}, g_{\mathcal{R}})$ where:*

- *M is a nonempty set called the* universe *of $\mathcal{M}$.*
- *$g_{\mathcal{C}} \colon \mathcal{C} \to M$.*
- *$g_{\mathcal{R}}$ is a function on $\mathcal{R}$ such that $g_{\mathcal{R}}(\mathsf{R})$ is a subset of M^k for all $\mathsf{R} \in \mathcal{R}_k$.*
- *$g_{\mathcal{F}}$ is a function on $\mathcal{F}$ such that $g_{\mathcal{F}}(\mathsf{f})$ is a k-ary function on M for all $\mathsf{f} \in \mathcal{F}_k$.*

We use the following notation:

- *For each $\mathsf{c} \in \mathcal{C}$, we use $\mathsf{c}^{\mathcal{M}}$ to denote $g_{\mathcal{C}}(\mathsf{c})$.*
- *For each $\mathsf{R} \in \mathcal{R}_k$, we use $\mathsf{R}^{\mathcal{M}}$ to denote $g_{\mathcal{R}}(\mathsf{R})$.*
- *For each $\mathsf{f} \in \mathcal{F}_k$, we use $\mathsf{f}^{\mathcal{M}}$ to denote $g_{\mathcal{F}}(\mathsf{f})$.*

For example, let $\mathcal{L}$ be the restricted group theory language, so $\mathcal{L} = \{\mathsf{e}, \mathsf{f}\}$, where e is a constant symbol and f is a binary function symbol. To give an $\mathcal{L}$-structure, we need to provide a set of elements M (to serve as the universe of discourse), pick an element of M to serve as the interpretation of e, and pick a function from M^2 to M to serve as the interpretation of f. Here are some examples of $\mathcal{L}$-structures:

1. $M = \mathbb{Z}$, $\mathsf{e}^{\mathcal{M}} = 3$ and $\mathsf{f}^{\mathcal{M}}$ is the subtraction function $(m, n) \mapsto m - n$ (in other words, $\mathsf{f}^{\mathcal{M}}(m, n) = m - n$).
2. $M = \mathbb{R}$, $\mathsf{e}^{\mathcal{M}} = \pi$ and $\mathsf{f}^{\mathcal{M}}$ is the function $(a, b) \mapsto \sin(a \cdot b)$.
3. For any group $(G, e, \cdot)$, we get an $\mathcal{L}$-structure by letting $M = G$, $\mathsf{e}^{\mathcal{M}} = e$, and letting $\mathsf{f}^{\mathcal{M}}$ be the group operation.

In particular, notice that in our restricted group theory language $\mathcal{L}$, there are $\mathcal{L}$-structures that are not groups! In other words, an $\mathcal{L}$-structure need not respect our intentions when writing down the symbols of $\mathcal{L}$. An $\mathcal{L}$-structure is *any* way to pick a set and a way to interpret the symbols as constants, relations, and functions. It is possible to carve out special collections of structures by only looking at those structures that make certain formulas true, but we first have to define "truth", as we will shortly.

For another example, let $\mathcal{L} = \{\mathsf{R}\}$ where R is a binary relation symbol. Here are some examples of $\mathcal{L}$-structures:

1. $M = \mathbb{N}$ and $\mathsf{R}^{\mathcal{M}} = \{(m,n) \in M^2 : m \mid n\}$.
2. $M = \{0,1\}^*$ and $\mathsf{R}^{\mathcal{M}} = \{(\sigma,\tau) \in M^2 : \sigma \preceq \tau\}$.
3. $M = \mathbb{R}^2$ and $\mathsf{R}^{\mathcal{M}} = \{((a_1,b_1),(a_2,b_2)) \in M^2 : a_1 = a_2\}$.
4. $M = \{0,1,2,3,4\}$ and $\mathsf{R}^{\mathcal{M}} = \{(0,2),(3,3),(4,1),(4,2),(4,3)\}$.

At first, it may appear than an $\mathcal{L}$-structure provides a means to make sense out of any formula. However, this is not the case, as we can see by looking at the formula $\mathsf{x} = \mathsf{y}$ where $\mathsf{x},\mathsf{y} \in Var$. Even given an $\mathcal{L}$-structure $\mathcal{M}$, we can't say whether the formula $\mathsf{x} = \mathsf{y}$ is "true" in $\mathcal{M}$ until we know how to interpret both x and y. For a more interesting example, consider the restricted group theory language $\mathcal{L} = \{\mathsf{e},\mathsf{f}\}$. Let $\mathcal{M}$ be the integers $\mathbb{Z}$ with $\mathsf{e}^{\mathcal{M}} = 0$ and with $\mathsf{f}^{\mathcal{M}}$ being addition. Consider the formula $\mathsf{fxy} = \mathsf{z}$. If we "interpret" x as 7, y as -3, and z as 4, then the formula $\mathsf{fxy} = \mathsf{z}$ is "true" in $\mathcal{M}$. However, if we "interpret" x as -2, y as 7, and z as 1, then the formula $\mathsf{fxy} = \mathsf{z}$ is "false" in $\mathcal{M}$. Once we fix an $\mathcal{L}$-structure $\mathcal{M}$, the need to interpret the elements of Var as elements of $\mathcal{M}$ motivates the following definition.

Definition 4.2.2. *Let $\mathcal{M}$ be an $\mathcal{L}$-structure. A function $s\colon Var \to M$ is called a* variable assignment on $\mathcal{M}$.

Recall that in propositional logic, every truth assignment $M\colon P \to \{0,1\}$ gave rise to a function $v_M\colon Form_P \to \{0,1\}$ telling us how to assign a true/false value to every formula. In the first-order logic case, we need an $\mathcal{L}$-structure $\mathcal{M}$ together with a variable assignment $s\colon Var \to M$ to make sense of things. We first explain how this apparatus allows us to assign an element of M to every term. We extend our function $s\colon Var \to M$ to a function $\overline{s}\colon Term_{\mathcal{L}} \to M$, similar to how we extend a truth assignment M to a function v_M. The distinction here is that s and $\overline{s}$ output elements of M rather than true/false values.

Definition 4.2.3. *Let $\mathcal{M}$ be an $\mathcal{L}$-structure, and let $s\colon Var \to M$ be a variable assignment. By freeness, there exists a unique $\overline{s}\colon Term_{\mathcal{L}} \to M$ with the following properties:*

- $\overline{s}(\mathsf{x}) = s(\mathsf{x})$ *for all* $\mathsf{v} \in Var$.
- $\overline{s}(\mathsf{c}) = \mathsf{c}^{\mathcal{M}}$ *for all* $\mathsf{c} \in \mathcal{C}$.
- $\overline{s}(\mathsf{f}t_1t_2\cdots t_k) = \mathsf{f}^{\mathcal{M}}(\overline{s}(t_1),\overline{s}(t_2),\ldots,\overline{s}(t_k))$.

Notice that there is nothing deep going on here. Given an $\mathcal{L}$-structure $\mathcal{M}$ and a variable assignment s, to apply $\overline{s}$ to a term, we simply unravel the term and attach "meaning" to each symbol (using $\mathcal{M}$ and s) as we bottom-out through the recursion. For example, assume that $\mathcal{L} = \{\mathsf{c},\mathsf{f}\}$ where c is a constant symbol and f is a binary function symbol. Given an $\mathcal{L}$-structure $\mathcal{M}$ and a variable assignment $s\colon Var \to M$, then working through the definitions, we have

$$\begin{aligned}
\overline{s}(\mathsf{ffczfxffczy}) &= \mathsf{f}^{\mathcal{M}}(\overline{s}(\mathsf{fcz}),\overline{s}(\mathsf{fxffczy})) \\
&= \mathsf{f}^{\mathcal{M}}(\mathsf{f}^{\mathcal{M}}(\overline{s}(\mathsf{c}),\overline{s}(\mathsf{z})),\mathsf{f}^{\mathcal{M}}(\overline{s}(\mathsf{x}),\overline{s}(\mathsf{ffczy}))) \\
&= \mathsf{f}^{\mathcal{M}}(\mathsf{f}^{\mathcal{M}}(\overline{s}(\mathsf{c}),\overline{s}(\mathsf{z})),\mathsf{f}^{\mathcal{M}}(\overline{s}(\mathsf{x}),\mathsf{f}^{\mathcal{M}}(\overline{s}(\mathsf{fcz}),\overline{s}(\mathsf{y})))) \\
&= \mathsf{f}^{\mathcal{M}}(\mathsf{f}^{\mathcal{M}}(\overline{s}(\mathsf{c}),\overline{s}(\mathsf{z})),\mathsf{f}^{\mathcal{M}}(\overline{s}(\mathsf{x}),\mathsf{f}^{\mathcal{M}}(\mathsf{f}^{\mathcal{M}}(\overline{s}(\mathsf{c}),\overline{s}(\mathsf{z})),\overline{s}(\mathsf{y})))) \\
&= \mathsf{f}^{\mathcal{M}}(\mathsf{f}^{\mathcal{M}}(\mathsf{c}^{\mathcal{M}},s(\mathsf{z})),\mathsf{f}^{\mathcal{M}}(s(\mathsf{x}),\mathsf{f}^{\mathcal{M}}(\mathsf{f}^{\mathcal{M}}(\mathsf{c}^{\mathcal{M}},s(\mathsf{z})),s(\mathsf{y}))))).
\end{aligned}$$

In other words, we're taking the *syntactic* formula $\mathsf{ffczfxffczy}$ and assigning a *semantic* meaning to it by returning the element of $\mathcal{M}$ described in the last line. For a specific example of how this would be interpreted, let $\mathcal{M}$ be the integers $\mathbb{Z}$ with $\mathsf{c}^{\mathcal{M}} = 5$ and with $\mathsf{f}^{\mathcal{M}}$ being addition. Let $s\colon Var \to M$ be an arbitrary variable assignment with $s(\mathsf{x}) = 3$, $s(\mathsf{y}) = -11$, and $s(\mathsf{z}) = 2$. We then have

$$\overline{s}(\mathsf{ffczfxffczy}) = 6$$

because

$$\begin{aligned}\overline{s}(\mathsf{ffczfxffczy}) &= \mathsf{f}^{\mathcal{M}}(\mathsf{f}^{\mathcal{M}}(\mathsf{c}^{\mathcal{M}}, s(\mathsf{z})), \mathsf{f}^{\mathcal{M}}(s(\mathsf{x}), \mathsf{f}^{\mathcal{M}}(\mathsf{f}^{\mathcal{M}}(\mathsf{c}^{\mathcal{M}}, s(\mathsf{z})), s(\mathsf{y}))))\\ &= \mathsf{f}^{\mathcal{M}}(\mathsf{f}^{\mathcal{M}}(0,2), \mathsf{f}^{\mathcal{M}}(3, \mathsf{f}^{\mathcal{M}}(\mathsf{f}^{\mathcal{M}}(0,2), -11)))\\ &= ((5+2)+(3+(5+2)+(-11)))\\ &= 6.\end{aligned}$$

We're now ready to define the intuitive statement "φ is true in the $\mathcal{L}$-structure $\mathcal{M}$ with variable assignment s" recursively. We need the following definition in order to handle quantifiers.

Definition 4.2.4. *Let $\mathcal{M}$ be an $\mathcal{L}$-structure, and let $s\colon Var \to M$ be a variable assignment. Given $\mathsf{x} \in Var$ and $a \in M$, we let $s[\mathsf{x} \Rightarrow a]$ denote the variable assignment*

$$s[\mathsf{x} \Rightarrow a](\mathsf{y}) = \begin{cases} a & \text{if } \mathsf{y} = \mathsf{x} \\ s(\mathsf{y}) & \text{otherwise.} \end{cases}$$

Now we can actually define the analogue of v_M from propositional logic. Given an $\mathcal{L}$-structure $\mathcal{M}$ and variable assignment $s\colon M \to Var$, we should be able to "make sense of" every $\varphi \in Form_{\mathcal{L}}$. In other words, we should be able to define a function $v_{(\mathcal{M},s)}\colon Form_{\mathcal{L}} \to \{0,1\}$, where the value 0 corresponds to false and the value 1 corresponds to true. Of course, the definition is recursive.

Definition 4.2.5. *Let $\mathcal{M}$ be an $\mathcal{L}$-structure. We recursively define a function $v_{(\mathcal{M},s)}\colon Form_P \to \{0,1\}$ for all $\varphi \in Form_{\mathcal{L}}$ and all variable assignments s as follows:*

- *We first handle the case where φ is an atomic formula:*
  - *If $\mathsf{R} \in \mathcal{R}_k$, and $t_1, t_2, \ldots, t_k \in Term_{\mathcal{L}}$, we let*

$$v_{(\mathcal{M},s)}(\mathsf{R}t_1t_2\ldots t_k) = \begin{cases} 1 & \text{if } (\overline{s}(t_1), \overline{s}(t_2), \ldots, \overline{s}(t_k)) \in \mathsf{R}^{\mathcal{M}} \\ 0 & \text{otherwise.} \end{cases}$$

  - *If $t_1, t_2 \in Term_{\mathcal{L}}$, we let*

$$v_{(\mathcal{M},s)}(= t_1t_2) = \begin{cases} 1 & \text{if } \overline{s}(t_1) = \overline{s}(t_2) \\ 0 & \text{otherwise.} \end{cases}$$

- *For any s, we let* $v_{(\mathcal{M},s)}(\neg\varphi) = \begin{cases} 1 & \text{if } v_{(\mathcal{M},s)}(\varphi) = 0 \\ 0 & \text{if } v_{(\mathcal{M},s)}(\varphi) = 1. \end{cases}$

- *For any s, we let* $v_{(\mathcal{M},s)}(\wedge\varphi\psi) = \begin{cases} 0 & \text{if } v_{(\mathcal{M},s)}(\varphi) = 0 \text{ and } v_{(\mathcal{M},s)}(\psi) = 0 \\ 0 & \text{if } v_{(\mathcal{M},s)}(\varphi) = 0 \text{ and } v_{(\mathcal{M},s)}(\psi) = 1 \\ 0 & \text{if } v_{(\mathcal{M},s)}(\varphi) = 1 \text{ and } v_{(\mathcal{M},s)}(\psi) = 0 \\ 1 & \text{if } v_{(\mathcal{M},s)}(\varphi) = 1 \text{ and } v_{(\mathcal{M},s)}(\psi) = 1. \end{cases}$

- *For any s, we let* $v_{(\mathcal{M},s)}(\vee\varphi\psi) = \begin{cases} 0 & \text{if } v_{(\mathcal{M},s)}(\varphi) = 0 \text{ and } v_{(\mathcal{M},s)}(\psi) = 0 \\ 1 & \text{if } v_{(\mathcal{M},s)}(\varphi) = 0 \text{ and } v_{(\mathcal{M},s)}(\psi) = 1 \\ 1 & \text{if } v_{(\mathcal{M},s)}(\varphi) = 1 \text{ and } v_{(\mathcal{M},s)}(\psi) = 0 \\ 1 & \text{if } v_{(\mathcal{M},s)}(\varphi) = 1 \text{ and } v_{(\mathcal{M},s)}(\psi) = 1. \end{cases}$

- *For any s, we let* $v_{(\mathcal{M},s)}(\to \varphi\psi) = \begin{cases} 1 & \text{if } v_{(\mathcal{M},s)}(\varphi) = 0 \text{ and } v_{(\mathcal{M},s)}(\psi) = 0 \\ 1 & \text{if } v_{(\mathcal{M},s)}(\varphi) = 0 \text{ and } v_{(\mathcal{M},s)}(\psi) = 1 \\ 0 & \text{if } v_{(\mathcal{M},s)}(\varphi) = 1 \text{ and } v_{(\mathcal{M},s)}(\psi) = 0 \\ 1 & \text{if } v_{(\mathcal{M},s)}(\varphi) = 1 \text{ and } v_{(\mathcal{M},s)}(\psi) = 1. \end{cases}$
- *For any s, we let* $v_{(\mathcal{M},s)}(\exists \mathsf{x}\varphi) = \begin{cases} 1 & \text{if there exists } a \in M \text{ with } v_{(\mathcal{M},s[\mathsf{x}\Rightarrow a])}(\varphi) = 1 \\ 0 & \text{otherwise.} \end{cases}$
- *For any s, we let* $v_{(\mathcal{M},s)}(\forall \mathsf{x}\varphi) = \begin{cases} 1 & \text{if for all } a \in M, \text{ we have } v_{(\mathcal{M},s[\mathsf{x}\Rightarrow a])}(\varphi) = 1 \\ 0 & \text{otherwise.} \end{cases}$

The above recursive definition takes a little explanation, because some recursive "calls" change the variable assignment. Thus, we are *not* fixing an $\mathcal{L}$-structure $\mathcal{M}$ and a variable assignment s on $\mathcal{M}$, and then doing a recursive definition on $\varphi \in Form_{\mathcal{L}}$. To fit this recursive definition into our framework from Chapter 2, we can adjust it as follows. Fix an $\mathcal{L}$-structure $\mathcal{M}$. Let $VarAssign_{\mathcal{M}}$ be the set of all variable assignments on $\mathcal{M}$. We then define a function $g_{\mathcal{M}} \colon Form_{\mathcal{P}} \to VarAssign_{\mathcal{M}}$ recursively using the above rules as guides, with the intention that $v_{(\mathcal{M},s)}(\varphi) = 1$ means that $s \in g_{\mathcal{M}}(\varphi)$. Here are three representative examples of how we define $g_{\mathcal{M}}$:

$$\begin{aligned} g_{\mathcal{M}}(\mathsf{R}t_1t_2\dots t_k) &= \{s \in VarAssign_{\mathcal{M}} : (\overline{s}(t_1), \overline{s}(t_2), \dots, \overline{s}(t_k)) \in \mathsf{R}^{\mathcal{M}}\} \\ g_{\mathcal{M}}(\wedge\varphi\psi) &= g_{\mathcal{M}}(\varphi) \cap g_{\mathcal{M}}(\psi) \\ g_{\mathcal{M}}(\exists \mathsf{x}\varphi) &= \bigcup_{a\in M} \{s \in VarAssign_{\mathcal{M}} : s[\mathsf{x} \Rightarrow a] \in g_{\mathcal{M}}(\varphi)\}. \end{aligned}$$

From here, we then *define* $v_{(\mathcal{M},s)}(\varphi) = 1$ to mean that $s \in g_{\mathcal{M}}(\varphi)$, and check that it has the required properties.

Let's consider a simple example. Let $\mathcal{L} = \{\mathsf{R}, \mathsf{f}\}$ where R is a unary relation symbol and f is a unary function symbol. Let $\mathcal{M}$ be the following $\mathcal{L}$-structure:

- $M = \{0, 1, 2, 3\}$.
- $\mathsf{R}^{\mathcal{M}} = \{1, 3\}$.
- $\mathsf{f}^{\mathcal{M}} \colon M \to M$ is the function defined by

$$\mathsf{f}^{\mathcal{M}}(0) = 3 \qquad \mathsf{f}^{\mathcal{M}}(1) = 1 \qquad \mathsf{f}^{\mathcal{M}}(2) = 0 \qquad \mathsf{f}^{\mathcal{M}}(3) = 3.$$

We now explore the values of $v_{(\mathcal{M},s)}(\varphi)$ for various choices of variable assignments s and formulas φ.

1. For any variable assignment $s \colon Var \to M$, we have

$$\begin{aligned} v_{(\mathcal{M},s)}(\neg\mathsf{R}\mathsf{x}) = 1 &\Leftrightarrow v_{(\mathcal{M},s)}(\mathsf{R}\mathsf{x}) = 0 \\ &\Leftrightarrow s(\mathsf{x}) \notin \mathsf{R}^{\mathcal{M}} \\ &\Leftrightarrow s(\mathsf{x}) = 0 \text{ or } s(\mathsf{x}) = 2. \end{aligned}$$

2. For any variable assignment $s \colon Var \to M$, we have

$$\begin{aligned} v_{(\mathcal{M},s)}(\exists \mathsf{x}\mathsf{R}\mathsf{x}) = 1 &\Leftrightarrow \text{There exists } a \in M \text{ such that } v_{(\mathcal{M},s[\mathsf{x}\Rightarrow a])}(\mathsf{R}\mathsf{x}) = 1 \\ &\Leftrightarrow \text{There exists } a \in M \text{ such that } \overline{s[\mathsf{x} \Rightarrow a]}(\mathsf{x}) \in \mathsf{R}^{\mathcal{M}} \\ &\Leftrightarrow \text{There exists } a \in M \text{ such that } s[\mathsf{x} \Rightarrow a](\mathsf{x}) \in \mathsf{R}^{\mathcal{M}} \\ &\Leftrightarrow \text{There exists } a \in M \text{ such that } a \in \mathsf{R}^{\mathcal{M}}. \end{aligned}$$

Since $\mathsf{R}^{\mathcal{M}} \neq \emptyset$, it follows that $v_{(\mathcal{M},s)}(\exists \mathsf{x}\mathsf{R}\mathsf{x}) = 1$ for all variable assignments $s \colon Var \to M$.

3. For any variable assignment $s\colon Var \to M$, we have

$$\begin{aligned} v_{(\mathcal{M},s)}(\forall \mathsf{x}(\mathsf{Rx} \to (\mathsf{fx} = \mathsf{x}))) = 1 &\Leftrightarrow \text{For all } a \in M, \text{ we have } v_{(\mathcal{M},s[\mathsf{x}\Rightarrow a])}((\mathsf{Rx} \to (\mathsf{fx} = \mathsf{x}))) = 1 \\ &\Leftrightarrow \text{For all } a \in M, \text{ we have either} \\ &\qquad v_{(\mathcal{M},s[\mathsf{x}\Rightarrow a])}(\mathsf{Rx}) = 0 \text{ or } v_{(\mathcal{M},s[\mathsf{x}\Rightarrow a])}(\mathsf{fx} = \mathsf{x}) = 1 \\ &\Leftrightarrow \text{For all } a \in M, \text{ we have either} \\ &\qquad \overline{s[\mathsf{x}\Rightarrow a]}(\mathsf{x}) \notin \mathsf{R}^{\mathcal{M}} \text{ or } \overline{s[\mathsf{x}\Rightarrow a]}(\mathsf{fx}) = \overline{s[\mathsf{x}\Rightarrow a]}(\mathsf{x}) \\ &\Leftrightarrow \text{For all } a \in M, \text{ we have either} \\ &\qquad s[\mathsf{x}\Rightarrow a](\mathsf{x}) \notin \mathsf{R}^{\mathcal{M}} \text{ or } \mathsf{f}^{\mathcal{M}}(\overline{s[\mathsf{x}\Rightarrow a]}(\mathsf{x})) = \overline{s[\mathsf{x}\Rightarrow a]}(\mathsf{x}) \\ &\Leftrightarrow \text{For all } a \in M, \text{ we have either} \\ &\qquad s[\mathsf{x}\Rightarrow a](\mathsf{x}) \notin \mathsf{R}^{\mathcal{M}} \text{ or } \mathsf{f}^{\mathcal{M}}(s[\mathsf{x}\Rightarrow a](\mathsf{x})) = s[\mathsf{x}\Rightarrow a](\mathsf{x}) \\ &\Leftrightarrow \text{For all } a \in M, \text{ we have either } a \notin \mathsf{R}^{\mathcal{M}} \text{ or } \mathsf{f}^{\mathcal{M}}(a) = a. \end{aligned}$$

   Since $0 \notin R^{\mathcal{M}}$, $\mathsf{f}^{\mathcal{M}}(1) = 1$, $2 \notin R^{\mathcal{M}}$, and $\mathsf{f}^{\mathcal{M}}(3) = 3$, it follows that $v_{(\mathcal{M},s)}(\forall \mathsf{x}(\mathsf{Rx} \to (\mathsf{fx} = \mathsf{x}))) = 1$ for all variable assignments $s\colon Var \to M$.

In the above examples, notice that only the values of s on the free variables in φ affected whether or not $v_{(\mathcal{M},s)} = 1$. In general, this seems intuitively clear, and we now state the corresponding precise result.

Proposition 4.2.6. *Let $\mathcal{M}$ be an $\mathcal{L}$-structure.*

1. *Suppose that $t \in Term_{\mathcal{L}}$ and $s_1, s_2\colon Var \to M$ are two variable assignments such that $s_1(\mathsf{x}) = s_2(\mathsf{x})$ for all $\mathsf{x} \in OccurVar(t)$. We then have $\overline{s}_1(t) = \overline{s}_2(t)$.*
2. *Let $\mathcal{M}$ be an $\mathcal{L}$-structure. Suppose that $\varphi \in Form_{\mathcal{L}}$ and $s_1, s_2\colon Var \to M$ are two variable assignments such that $s_1(\mathsf{x}) = s_2(\mathsf{x})$ for all $\mathsf{x} \in FreeVar(\varphi)$. We then have*

$$v_{(\mathcal{M},s_1)}(\varphi) = 1 \text{ if and only if } v_{(\mathcal{M},s_2)}(\varphi) = 1.$$

Proof. Each of these is a straightforward induction, the first on $t \in Term_{\mathcal{L}}$ and the second on $\varphi \in Form_{\mathcal{L}}$. □

We introduced the notation $v_{(\mathcal{M},s)}(\varphi)$ to correspond to our old notation $v_M(\varphi)$. In propositional logic, we needed a truth assignment $M\colon P \to \{0,1\}$ to assign true/false values to all formulas. In first-order logic, we need both an $\mathcal{L}$-structure $\mathcal{M}$ and a variable assignment $s\colon Var \to M$ to assign true/false values to all formulas. Despite the advantages of the similar notation, it is tiresome to keep writing so much in the subscripts, and so people who work in mathematical logic have adopted other standard notation, which we now introduce.

Notation 4.2.7. *Let $\mathcal{L}$ be a language, let $\mathcal{M}$ be an $\mathcal{L}$-structure, let $s\colon Var \to M$ be a variable assignment, and let $\varphi \in Form_{\mathcal{L}}$. We write $(\mathcal{M}, s) \vDash \varphi$ to mean that $v_{(\mathcal{M},s)}(\varphi) = 1$, and write $(\mathcal{M}, s) \nvDash \varphi$ to mean that $v_{(\mathcal{M},s)}(\varphi) = 0$.*

In some ways, using the symbol $\vDash$ is natural, because we are defining the semantic notion that φ is *true* in $(\mathcal{M}, s)$. However, in other ways, this notation is incredibly confusing. In propositional logic, we used the symbol $\vDash$ only in $\Gamma \vDash \varphi$ where $\Gamma \subseteq Form_P$ and $\varphi \in Form_P$. In other words, we used $\vDash$ for *semantic implication*, not semantic truth. Our new first-order logic notation would be akin to also writing $M \vDash \varphi$ in propositional logic to mean that $v_M(\varphi) = 1$. Although we could have done that in the Chapter 3, we avoided the temptation to overload the notation at that stage. We will eventually define $\Gamma \vDash \varphi$ in first-order logic when $\Gamma \subseteq Form_{\mathcal{L}}$ and $\varphi \in Form_{\mathcal{L}}$, and at that point we will just have to know which version of $\vDash$ we are using based on what type of object appears on the left. Consider yourself warned!

With this new notation in hand, we can rewrite the recursive definition of $v_{(\mathcal{M},s)}$ in the following way:

- Suppose first that φ is an atomic formula.
  - If φ is $\mathsf{R}t_1t_2\cdots t_k$, we have $(\mathcal{M}, s) \vDash \varphi$ if and only if $(\overline{s}(t_1), \overline{s}(t_2), \ldots, \overline{s}(t_k)) \in \mathsf{R}^{\mathcal{M}}$.
  - If φ is $= t_1t_2$, we have $(\mathcal{M}, s) \vDash \varphi$ if and only if $\overline{s}(t_1) = \overline{s}(t_2)$.
- For any s, we have $(\mathcal{M}, s) \vDash \neg\varphi$ if and only if $(\mathcal{M}, s) \nvDash \varphi$.
- For any s, we have $(\mathcal{M}, s) \vDash \varphi \wedge \psi$ if and only if $(\mathcal{M}, s) \vDash \varphi$ and $(\mathcal{M}, s) \vDash \psi$.
- For any s, we have $(\mathcal{M}, s) \vDash \varphi \vee \psi$ if and only if either $(\mathcal{M}, s) \vDash \varphi$ or $(\mathcal{M}, s) \vDash \psi$.
- For any s, we have $(\mathcal{M}, s) \vDash \varphi \to \psi$ if and only if either $(\mathcal{M}, s) \nvDash \varphi$ or $(\mathcal{M}, s) \vDash \psi$.
- For any s, we have $(\mathcal{M}, s) \vDash \exists \mathsf{x}\varphi$ if and only if there exists $a \in M$ such that $(\mathcal{M}, s[\mathsf{x} \Rightarrow a]) \vDash \varphi$.
- For any s, we have $(\mathcal{M}, s) \vDash \forall \mathsf{x}\varphi$ if and only if for all $a \in M$, we have $(\mathcal{M}, s[\mathsf{x} \Rightarrow a]) \vDash \varphi$.

We now introduce some other notation in light of Proposition 4.2.6.

Notation 4.2.8. *Let $\mathcal{L}$ be a language.*

1. *If $\mathsf{x}_1, \mathsf{x}_2, \ldots, \mathsf{x}_n \in Var$ are distinct, and we refer to a formula $\varphi(\mathsf{x}_1, \mathsf{x}_2, \ldots, \mathsf{x}_n) \in Form_{\mathcal{L}}$ we mean that $\varphi \in Form_{\mathcal{L}}$ and $FreeVar(\varphi) \subseteq \{\mathsf{x}_1, \mathsf{x}_2, \ldots, \mathsf{x}_n\}$.*
2. *Suppose that $\mathcal{M}$ is an $\mathcal{L}$-structure, $\varphi(\mathsf{x}_1, \mathsf{x}_2, \ldots, \mathsf{x}_n) \in Form_{\mathcal{L}}$, and $a_1, a_2, \ldots, a_n \in M$. We write $(\mathcal{M}, a_1, a_2, \ldots, a_n) \vDash \varphi$ to mean that $(\mathcal{M}, s) \vDash \varphi$ for some (any) $s\colon Var \to M$ with $s(\mathsf{x}_i) = a_i$ for all i.*
3. *As a special case of 2, we have the following. Suppose that $\mathcal{M}$ is an $\mathcal{L}$-structure and $\sigma \in Sent_{\mathcal{L}}$. We write $\mathcal{M} \vDash \sigma$ to mean that $(\mathcal{M}, s) \vDash \sigma$ for some (any) $s\colon Var \to M$.*

As we've seen, given a language $\mathcal{L}$, an $\mathcal{L}$-structure can be any set M together with any interpretation of the symbols. In particular, although we might only have certain structures in mind when we describe a language, the structures themselves need not respect our desires. However, since we have now formally defined the intuitive notion that a sentence $\varphi \in Sent_{\mathcal{L}}$ is true in an $\mathcal{L}$-structure $\mathcal{M}$ (recall that a sentence has no free variables, so we don't need a variable assignment), we now carve out classes of structures which satisfy certain sentences of our language.

Definition 4.2.9. *Let $\mathcal{L}$ be a language, and let $\Sigma \subseteq Sent_{\mathcal{L}}$. We let $Mod(\Sigma)$ be the class of all $\mathcal{L}$-structures $\mathcal{M}$ such that $\mathcal{M} \vDash \sigma$ for all $\sigma \in \Sigma$. If $\sigma \in Sent_{\mathcal{L}}$, we write $Mod(\sigma)$ instead of $Mod(\{\sigma\})$.*

Definition 4.2.10. *Let $\mathcal{L}$ be a language and let $\mathcal{K}$ be a class of $\mathcal{L}$-structures.*

1. *$\mathcal{K}$ is an* elementary class *if there exists $\Sigma \subseteq Sent_{\mathcal{L}}$ such that $\mathcal{K} = Mod(\Sigma)$.*
2. *$\mathcal{K}$ is a* strong elementary class *if there exists $\sigma \in Sent_{\mathcal{L}}$ such that $\mathcal{K} = Mod(\sigma)$.*

By taking conjunctions, we have the following simple proposition.

Proposition 4.2.11. *Let $\mathcal{L}$ be a language and let $\mathcal{K}$ be a class of $\mathcal{L}$-structures. $\mathcal{K}$ is a strong elementary class if and only if there exists a finite $\Sigma \subseteq Sent_{\mathcal{L}}$ such that $\mathcal{K} = Mod(\Sigma)$.*

For example, if we let $\mathcal{L} = \{\mathsf{e}, \mathsf{f}\}$ be the restricted group theory language, then the class of groups is a strong elementary class as we saw in Chapter 1, because we can let Σ be the following collection of sentences:

(a) $\forall \mathsf{x}\forall \mathsf{y}\forall \mathsf{z}(\mathsf{f}(\mathsf{f}(\mathsf{x}, \mathsf{y}), \mathsf{z}) = \mathsf{f}(\mathsf{x}, \mathsf{f}(\mathsf{y}, \mathsf{z})))$.

(b) $\forall \mathsf{x}((\mathsf{f}(\mathsf{x}, \mathsf{e}) = \mathsf{x}) \wedge (\mathsf{f}(\mathsf{e}, \mathsf{x}) = \mathsf{x}))$.

(c) $\forall x \exists y((f(x,y) = e) \wedge (f(y,x) = e))$.

If we instead use the (full) group theory language $\mathcal{L} = \{e, f, g\}$, then the class of groups is a strong elementary class by letting Σ be the following collection of sentences:

(a) $\forall x \forall y \forall z(f(f(x,y),z) = f(x,f(y,z)))$.

(b) $\forall x((f(x,e) = x) \wedge (f(e,x) = x))$.

(c) $\forall x((f(x,g(x)) = e) \wedge (f(g(x),x) = e))$.

Moving on to another language, consider $\mathcal{L} = \{R\}$ where R is a binary relation symbol. We can form several fundamental strong elementary classes in this language:

1. The class of partially ordered sets is a strong elementary class as we saw in Chapter 1, because we can let Σ be the following collection of sentences:
   (a) $\forall x Rxx$.
   (b) $\forall x \forall y((Rxy \wedge Ryx) \rightarrow (x = y))$.
   (c) $\forall x \forall y \forall z((Rxy \wedge Ryz) \rightarrow Rxz)$.
2. The class of equivalence relations is a strong elementary class by letting Σ be the following collection of sentences:
   (a) $\forall x Rxx$.
   (b) $\forall x \forall y(Rxy \rightarrow Ryx)$.
   (c) $\forall x \forall y \forall z((Rxy \wedge Ryz) \rightarrow Rxz)$.
3. The class of simple undirected graphs (i.e. edges have no direction, and there are no loops and no multiple edges) is a strong elementary class by letting Σ be the following collection of sentences:
   (a) $\forall x(\neg Rxx)$.
   (b) $\forall x \forall y(Rxy \rightarrow Ryx)$.

For another example, let $\mathcal{L} = \{0, 1, +, \cdot, -\}$ where $0, 1$ are constant symbols and $+, \cdot$ are binary function symbols, and $-$ is a unary function symbol. We call $\mathcal{L}$ the *ring theory language*. As in our group theory language, we are including a unary function symbol for additive inverses for reasons that will be explained in Section 4.3

1. The class of (possibly noncommutative) rings with identity is a strong elementary class by letting Σ be the following collection of sentences:
   (a) $\forall x \forall y \forall z(x + (y + z) = (x + y) + z)$.
   (b) $\forall x((x + 0 = x) \wedge (0 + x = x))$.
   (c) $\forall x((x + (-x) = 0) \wedge ((-x) + x = 0))$.
   (d) $\forall x \forall y(x + y = y + x)$.
   (e) $\forall x((x \cdot 1 = x) \wedge (1 \cdot x = x))$.
   (f) $\forall x \forall y \forall z(x \cdot (y \cdot z) = (x \cdot y) \cdot z)$.
   (g) $\forall x \forall y \forall z(x \cdot (y + z) = (x \cdot y) + (x \cdot z))$.
2. The class of fields is a strong elementary class by letting Σ be the above sentences together with:

(h) $\neg(0 = 1)$.
(i) $\forall x \forall y(x \cdot y = y \cdot x)$.
(j) $\forall x(\neg(x = 0) \to \exists y(x \cdot y = 1))$.

3. For each prime $p > 0$, the class of fields of characteristic p is a strong elementary class. To see this, fix a prime $p > 0$, and let Σ_p be the above sentences together with the sentence $1 + 1 + \cdots + 1 = 0$ (where there are p many 1's in the sum).

4. The class of fields of characteristic 0 is an elementary class because we can let Σ be the above sentences together with $\{\tau_n : n \in \mathbb{N}^+\}$ where for each $n \in \mathbb{N}^+$, we have $\tau_n = \neg(1 + 1 + \cdots + 1 = 0)$ (where there are n many 1's in the sum).

We now consider an example with an infinite language. Let F be a field, and let $\mathcal{L}_F = \{0, +\} \cup \{h_\alpha : \alpha \in F\}$ where 0 is a constant symbol, $+$ is binary function symbol, and each h_α is a unary function symbol. The class of vector spaces over F is an elementary class, by letting Σ be the following collection of sentences:

1. $\forall x \forall y \forall z(x + (y + z) = (x + y) + z)$.
2. $\forall x((x + 0 = x) \wedge (0 + x = x))$.
3. $\forall x \exists y((x + y = 0) \wedge (y + x = 0))$.
4. $\forall x \forall y(x + y = y + x)$.
5. $\forall x \forall y(h_\alpha(x + y) = h_\alpha(x) + h_\alpha(y))$ for each $\alpha \in F$.
6. $\forall x(h_{\alpha+\beta}(x) = (h_\alpha(x) + h_\beta(x)))$ for each $\alpha, \beta \in F$.
7. $\forall x(h_{\alpha\cdot\beta}(x) = h_\alpha(h_\beta(x)))$ for each $\alpha, \beta \in F$.
8. $\forall x(h_1(x) = x)$.

Notice that if F is infinite, then we really have infinitely many formulas here, because three of the sentences are parametrized by elements of F.

Finally, notice that given *any* language $\mathcal{L}$ and any $n \in \mathbb{N}^+$, the class of $\mathcal{L}$-structures of cardinality at least n is a strong elementary class as witnessed by the formula

$$\exists x_1 \exists x_2 \cdots \exists x_n \left(\bigwedge_{1 \le i < j \le n} \neg(x_i = x_j) \right).$$

Furthermore, the class of $\mathcal{L}$-structures of cardinality equal to n is a strong elementary class. To see this, let σ_n be the above formula for n, and consider the sentence $\sigma_n \wedge (\neg\sigma_{n+1})$.

At this point, it's often clear how to show that a certain class of structures is a (strong) elementary class: simply exhibit a correct set of sentences. However, it is unclear how one could show that a class is *not* a (strong) elementary class. For example, is the class of fields of characteristic 0 a strong elementary class? Is the class of finite groups an elementary class? There are no obvious ways to answer these questions affirmatively. We will eventually develop tools that will allow us to resolve these questions negatively.

Another interesting case is the class of Dedekind-complete ordered fields. Now the ordered field axioms are easily written down in the first-order language $\mathcal{L} = \{0, 1, +, -, \cdot, <\}$ of ring theory augmented by a binary relation symbol $<$. In contrast, the Dedekind-completeness axiom, which says that every nonempty subset that is bounded above has a least upper bound, can not be directly translated in the language $\mathcal{L}$ because it involves quantifying over subsets instead of elements. However, we are unable to immediately conclude that this isn't due to a lack of cleverness on our part. Perhaps there is an alternative approach which captures Dedekind-complete ordered fields in a first-order way (by finding a clever equivalent first-order expression of Dedekind-completeness). More formally, the precise question is whether the complete ordered fields is an elementary class in the language $\mathcal{L}$. We'll be able to answer this question in the negative later as well.

4.3 Substructures and Homomorphisms

One of the basic ways to obtain new algebraic structures (whether vector spaces, groups, or rings) is to find them "inside" already established ones. For example, when working in the vector space of all functions $f\colon \mathbb{R} \to \mathbb{R}$ under the usual pointwise operations, we can form the subspace of continuous functions, or the subspace of differentiable functions. Within the symmetric groups S_n, one naturally defines the alternating groups A_n and dihedral groups D_n. These ideas also arise in combinatorial structures, such as when examining subgraphs of a given graph, or viewing a partial ordering as a piece of a larger one. The general unifying concept here is that of a substructure.

Definition 4.3.1. *Let $\mathcal{L}$ be a language and let $\mathcal{M}$ and $\mathcal{A}$ be $\mathcal{L}$-structures. We say that $\mathcal{A}$ is a* substructure *of $\mathcal{M}$ if the following conditions hold:*

1. *$A \subseteq M$, where A and M are the underlying sets of $\mathcal{A}$ and $\mathcal{M}$, respectively.*
2. *$\mathsf{c}^{\mathcal{A}} = \mathsf{c}^{\mathcal{M}}$ for all $\mathsf{c} \in \mathcal{C}$.*
3. *$\mathsf{R}^{\mathcal{A}} = \mathsf{R}^{\mathcal{M}} \cap A^k$ for all $\mathsf{R} \in \mathcal{R}_k$.*
4. *$\mathsf{f}^{\mathcal{A}} = \mathsf{f}^{\mathcal{M}} \restriction A^k$ for all $\mathsf{f} \in \mathcal{F}_k$.*

In other words, a structure $\mathcal{A}$ is a substructure of $\mathcal{M}$ if we may have thrown away some of the elements of the set M, but on the remaining set A we have faithfully maintained the interpretation of every symbol. Notice that the second and fourth conditions imply a few simple facts about our set $A \subseteq M$. We prove that these are necessary and sufficient conditions for A to be the universe (i.e. the underlying set) of a substructure of $\mathcal{M}$.

Proposition 4.3.2. *Let $\mathcal{M}$ be an $\mathcal{L}$-structure, and let $A \subseteq M$ be nonempty. The following are equivalent:*

1. *A is the universe of a substructure of M, i.e. there is a substructure $\mathcal{A}$ of $\mathcal{M}$ with A as the underlying set.*
2. *Every element of M that is named by a constant must appear in A, and A is closed under every function $\mathsf{f}^{\mathcal{M}}$. More formally, we have $\{\mathsf{c}^{\mathcal{M}} : \mathsf{c} \in \mathcal{C}\} \subseteq A$ and $\mathsf{f}^{\mathcal{M}}(a_1, a_2, \ldots, a_k) \in A$ for all $\mathsf{f} \in \mathcal{F}_k$ and all $a_1, a_2, \ldots, a_k \in A$.*

Proof. We first prove that (1) implies (2). Let $\mathcal{A}$ be a substructure of $\mathcal{M}$ with underlying set A. For any $\mathsf{c} \in \mathcal{C}$, we have $\mathsf{c}^{\mathcal{A}} \in A$ by definition of a structure, and $\mathsf{c}^{\mathcal{A}} = \mathsf{c}^{\mathcal{M}}$ by definition of a substructure, so we conclude that $\mathsf{c}^{\mathcal{M}} \in A$. Now let $f \in \mathcal{F}_k$ and $a_1, a_2, \ldots, a_k \in A$ be arbitrary. We have $f^{\mathcal{A}}(a_1, a_2, \ldots, a_k) \in A$ by definition of a structure, so since $\mathsf{f}^{\mathcal{A}} = \mathsf{f}^{\mathcal{M}} \restriction A^k$, we conclude that $f^{\mathcal{M}}(a_1, a_2, \ldots, a_k) = f^{\mathcal{A}}(a_1, a_2, \ldots, a_k) \in A$.

We now prove that (2) implies (1). Assume then that $\{\mathsf{c}^{\mathcal{M}} : \mathsf{c} \in \mathcal{C}\} \subseteq A$ and $\mathsf{f}^{\mathcal{M}}(a_1, a_2, \ldots, a_k) \in A$ for all $\mathsf{f} \in \mathcal{F}_k$ and all $a_1, a_2, \ldots, a_k \in A$. We can then define each $\mathsf{c}^{\mathcal{A}}$, $\mathsf{R}^{\mathcal{A}}$, and $\mathsf{f}^{\mathcal{A}}$ as in Definition 4.3.1, and notice that these definitions make sense by our assumptions (i.e. we have each $\mathsf{c}^{\mathcal{A}} \in A$ and each $\mathsf{f}^{\mathcal{A}}$ is actually a function from A^k to A). Therefore, A is the universe of a substructure of $\mathcal{A}$. □

For example, suppose that we are working in the restricted group theory language $\mathcal{L} = \{\mathsf{e}, \mathsf{f}\}$, and we let $\mathcal{M}$ be the $\mathcal{L}$-structure with universe $\mathbb{Z}$, $\mathsf{e}^{\mathcal{M}} = 0$, and $\mathsf{f}^{\mathcal{M}}$ equal to the usual addition. We then have that $\mathcal{M}$ is a group. Notice that if we let $A = \mathbb{N}$, then A contains $0 = \mathsf{e}^{\mathcal{M}}$ and is closed under $\mathsf{f}^{\mathcal{M}}$. In other words, we can view A as the universe of a substructure $\mathcal{A}$ of $\mathcal{M}$. However, notice that $\mathcal{A}$ is *not* a subgroup of $\mathcal{M}$, because it is not closed under inverses. The problem here is that the inverse function is not the interpretation of any of the function symbols, so a substructure need not be closed under it. However, if we use the (full) group theory language where we include a unary function that is interpreted as the inverse function, then a substructure is precisely the same thing as a subgroup. In other words, our choice of language affects what

our substructures are. The fact that substructures correspond to subgroups in the latter language is the reason why we prefer to use it when working with group theory.

A similar situation happens with the choice of a ring theory language. If we work with the language $\mathcal{L} = \{0, 1, +, \cdot\}$, and we have an $\mathcal{L}$-structure $\mathcal{M}$ that is a ring, then a substructure would necessarily contain both the additive and multiplicative identities, and be closed under both addition and multiplication. However, these conditions do not force a substructure to be closed under additive inverses (again consider $\mathbb{N}$ as a subset of the ring $\mathbb{Z}$), and so substructures do not correspond to subrings. Thus, if we want substructures to be subrings, then we should include a unary function symbol $-$ for additive inverses (or, alternatively, a binary function $-$ for subtraction). Notice that because we include the constant symbol $\mathsf{1}$, every substructure of a ring will be a subring with the *same* multiplicative identity, which contrasts with the way a few (unenlightened) sources define subrings. Finally, note that even if our $\mathcal{L}$-structure happened to be a field, then a substructure need *not* be a subfield, but merely a subring.

Our discussion shows that we can often change the language in order to make substructures correspond to the usual definition of "subobject" in a given area mathematics. However, there are some situations where this is not possible with our definition of substructure. To see this, note that in our definition, an element of A^k is an element of $\mathsf{R}^{\mathcal{A}}$ if and only if it is an element of $\mathsf{R}^{\mathcal{M}}$, and this condition does not always match up with standard definitions of "subobjects". For example, suppose we are working in the language $\mathcal{L} = \{\mathsf{R}\}$, where R is a binary relation symbol. Suppose that $\mathcal{M}$ is an $\mathcal{L}$-structure that is a graph, i.e. such that $\mathcal{M} \vDash \forall \mathsf{x} \forall \mathsf{y}(\mathsf{Rxy} \to \mathsf{Ryx})$. Typically, a subgraph of a given graph is a subset where we are allowed to omit some vertices and edges, including the possibility of removing an edge despite keeping its endpoints. For example, if $M = \{1, 2, 3, 4\}$ and

$$\mathsf{R}^{\mathcal{M}} = \{(1, 2), (2, 1), (1, 3), (3, 1), (2, 3), (3, 2), (3, 4), (4, 3)\},$$

then the graph with $A = \{1, 2, 3\}$ and

$$\mathsf{R}^{\mathcal{A}} = \{(1, 2), (2, 1), (1, 3), (3, 1)\}$$

is a subgraph under the standard definition. Notice, however, that the corresponding $\mathcal{A}$ is *not* a substructure of $\mathcal{M}$, because

$$\mathsf{R}^{\mathcal{M}} \restriction A^2 = \{(1, 2), (2, 1), (1, 3), (3, 1), (2, 3), (3, 2)\}.$$

In this context, the definition of a substructure matches up with the concept of an *induced* subgraph. In order to handle situations like general subgraphs, some sources define the concept of a *weak substructure* by leaving conditions (1), (3), and (4) of Definition 4.3.1 alone, but changing condition (2) to be $\mathsf{R}^{\mathcal{A}} \subseteq \mathsf{R}^{\mathcal{M}}$ for all $\mathsf{R} \in \mathcal{R}$.

Suppose that we have a structure $\mathcal{M}$, and we have an arbitrary set $B \subseteq M$. How can we get a substructure $\mathcal{A}$ of $\mathcal{M}$ such that $B \subseteq A$? By Proposition 4.3.2, we need to ensure that A contains both B and the interpretation of the constants, and is closed under the functions $\mathsf{f}^{\mathcal{M}}$. Thus, we can build a substructure (in fact the smallest substructure containing B) by starting with B and the various $\mathsf{c}^{\mathcal{M}}$, and then generating new elements. In other words, we have the following result. Notice that we need to assume that either $B \neq \emptyset$ or $\mathcal{C} \neq \emptyset$ so that we start with a nonempty set, since we require that $A \neq \emptyset$ (because the underlying set of *any* structure must be nonempty by definition).

Corollary 4.3.3. *Let $\mathcal{M}$ be an $\mathcal{L}$-structure and let $B \subseteq M$ be an arbitrary set. Suppose either that $B \neq \emptyset$ or $\mathcal{C} \neq \emptyset$. If we let $A = G(M, B \cup \{\mathsf{c}^{\mathcal{M}} : \mathsf{c} \in \mathcal{C}\}, \{\mathsf{f}^{\mathcal{M}} : \mathsf{f} \in \mathcal{F}\})$, then A is the universe of a substructure of $\mathcal{M}$. Moreover, if $\mathcal{N}$ is any substructure of $\mathcal{M}$ with $B \subseteq N$, then $A \subseteq N$.*

We now seek to generalize the concept of a homomorphism from its algebraic roots to more general structures. In group theory, a homomorphism is a function that preserves the only binary operation. In this setting, it is straightforward to check that a group homomorphism automatically sends the identity to the identity, and sends the inverse of an element to the inverse of the image of that element. For rings

with identity, it is possible to have a function that preserves both addition and multiplication, but not the multiplicative identity. As a result, some sources enforce the additional condition that a ring homomorphism must also preserve the multiplicative identity. As in the definition of substructure, we require that the interpretation of the each of the first-order symbols from $\mathcal{L}$ is preserved. In other words, if we choose to include a constant symbol for the multiplicative identity in our language, then the multiplicative identity of the first ring must be sent to the multiplicative identity of the second. Thus, as with substructures, our choice of language will affect which functions we call homomorphisms.

Definition 4.3.4. *Let $\mathcal{L}$ be a language, and let $\mathcal{M}$ and $\mathcal{N}$ be $\mathcal{L}$-structures.*

1. *A function $h\colon M \to N$ is called a* homomorphism *if it satisfies the following conditions:*
   (a) *For all $\mathsf{c} \in \mathcal{C}$, we have $h(\mathsf{c}^{\mathcal{M}}) = \mathsf{c}^{\mathcal{N}}$.*
   (b) *For all $\mathsf{R} \in \mathcal{R}_k$ and all $a_1, a_2, \ldots, a_k \in M$, we have*
   $$(a_1, a_2, \ldots, a_k) \in \mathsf{R}^{\mathcal{M}} \text{ if and only if } (h(a_1), h(a_2), \ldots, h(a_k)) \in \mathsf{R}^{\mathcal{N}}.$$
   (c) *For all $f \in \mathcal{F}_k$ and all $a_1, a_2, \ldots, a_k \in M$, we have*
   $$h(\mathsf{f}^{\mathcal{M}}(a_1, a_2, \ldots, a_k)) = \mathsf{f}^{\mathcal{N}}(h(a_1), h(a_2), \ldots, h(a_k)).$$
2. *A function $h\colon M \to N$ is called an* embedding *if it is an injective homomorphism.*
3. *A function $h\colon M \to N$ is called an* isomorphism *if it is a bijective homomorphism.*

We pause again to note that this definition occasionally differs from the usual mathematical definition of homomorphism in languages with relation symbols. For instance, a graph homomorphism is often defined as a function with the property that if (a, b) is an edge in the first graph, then $(h(a), h(b))$ is an edge in the second. Note that this definition corresponds to only one direction of (b) above, rather than an if and only if. Some logic sources do indeed change (b) in the definition of homomorphism to the weaker condition that whenever $(a_1, a_2, \ldots, a_k) \in \mathsf{R}^{\mathcal{M}}$, we have $(h(a_1), h(a_2), \ldots, h(a_k)) \in \mathsf{R}^{\mathcal{N}}$. Such sources often use the terminology *strong homomorphism* for what we are simply calling a homomorphism, while sources that use our definition of homomorphism sometimes refer to the weaker notion as a *positive homomorphism*. In either case, embeddings and isomorphisms are always assumed to satisfy the if and only if described in part (b) above, so there is no ambiguity when using these terms. Since we will almost exclusively work with embeddings and isomorphisms (rather than just homomorphisms), and because our choice simplifies the statement of part (3) of Theorem 4.3.6, we choose to adopt the above version of the definition of homomorphism for simplicity. Just note that part (3) of Theorem 4.3.6 below would need to be modified if one were to adopt the weaker definition of homomorphism.

Proposition 4.3.5. *Let $\mathcal{L}$ be a language and let $\mathcal{M}$ and $\mathcal{A}$ be $\mathcal{L}$-structures with $A \subseteq M$. We then have that $\mathcal{A}$ is a substructure of $\mathcal{M}$ if and only if the inclusion function $i\colon A \to M$ given by $i(a) = a$ is an embedding.*

Proof. Immediate from the corresponding definitions. □

We now jump into our primary theorem about homomorphisms and isomorphisms. The last part of this theorem gives one precise way to say that all "reasonable" properties are preserved by an isomorphism. It's not an at all clear how to make this precise in algebra, but now we have a formal language that allows us to codify (at least some) "reasonable" properties. Since first-order formulas are generated from atomic formulas using simple rules, we can prove by induction that all properties expressible by first-order formulas are preserved.

Theorem 4.3.6. *Let $\mathcal{L}$ be a language, and let $\mathcal{M}$ and $\mathcal{N}$ be $\mathcal{L}$-structures. Suppose that $h\colon M \to N$ is a homomorphism, and suppose that $s\colon Var \to M$ is a variable assignment. We have the following:*

1. *The function $h \circ s$ is a variable assignment on $\mathcal{N}$.*
2. *For any $t \in Term_{\mathcal{L}}$, we have $h(\overline{s}(t)) = \overline{h \circ s}(t)$.*
3. *For every quantifier-free $\varphi \in Form_{\mathcal{L}}$ not containing the equality symbol, i.e. for all φ generated by starting only with atomic formulas using elements of $\mathcal{R}$, and generating with just the propositional connectives, we have*
$$(\mathcal{M}, s) \vDash \varphi \text{ if and only if } (\mathcal{N}, h \circ s) \vDash \varphi.$$
4. *If h is an embedding, then for every quantifier-free $\varphi \in Form_{\mathcal{L}}$, i.e. for all φ generated by starting with all atomic formulas and using just the propositional connectives, we have*
$$(\mathcal{M}, s) \vDash \varphi \text{ if and only if } (\mathcal{N}, h \circ s) \vDash \varphi.$$
5. *If h is an isomorphism, then for every $\varphi \in Form_{\mathcal{L}}$, we have*
$$(\mathcal{M}, s) \vDash \varphi \text{ if and only if } (\mathcal{N}, h \circ s) \vDash \varphi.$$

Proof.

1. This is immediate from the fact that $s\colon Var \to M$ and $h\colon M \to N$, so $h \circ s\colon Var \to N$.
2. The proof is by induction on t. We have two base cases. For any $\mathsf{c} \in \mathcal{C}$, we have
$$\begin{aligned} h(\overline{s}(\mathsf{c})) &= h(\mathsf{c}^{\mathcal{M}}) \\ &= \mathsf{c}^{\mathcal{N}} && \text{(since } h \text{ is a homomorphism)} \\ &= \overline{h \circ s}(\mathsf{c}). \end{aligned}$$
Also, for any $\mathsf{x} \in Var$, we have
$$\begin{aligned} h(\overline{s}(\mathsf{x})) &= h(s(\mathsf{x})) \\ &= (h \circ s)(\mathsf{x}) \\ &= \overline{h \circ s}(\mathsf{x}). \end{aligned}$$
For the inductive step, let $\mathsf{f} \in \mathcal{F}_k$ and let $t_1, t_2, \ldots, t_k \in Term_{\mathcal{L}}$ be such that the statement is true for each t_i. We then have
$$\begin{aligned} h(\overline{s}(\mathsf{f}t_1t_2\cdots t_k)) &= h(\mathsf{f}^{\mathcal{M}}(\overline{s}(t_1), \overline{s}(t_2), \ldots, \overline{s}(t_k))) && \text{(by definition of } \overline{s}) \\ &= \mathsf{f}^{\mathcal{M}}(h(\overline{s}(t_1)), h(\overline{s}(t_2)), \ldots, h(\overline{s}(t_k))) && \text{(since } h \text{ is a homomorphism)} \\ &= \mathsf{f}^{\mathcal{N}}(\overline{h \circ s}(t_1), \overline{h \circ s}(t_2), \ldots, \overline{h \circ s}(t_k)) && \text{(by induction)} \\ &= \overline{h \circ s}(\mathsf{f}t_1t_2\cdots t_k) && \text{(by definition of } \overline{h \circ s}). \end{aligned}$$
Hence, the statement is true for $\mathsf{f}t_1t_2\cdots t_k$, which completes the induction.
3. Assume that h is an embedding. The proof is by induction on φ. Let $\mathsf{R} \in \mathcal{R}_k$ and $t_1, t_2, \ldots, t_k \in Term_{\mathcal{L}}$ be arbitrary. We have
$$\begin{aligned} (\mathcal{M}, s) \vDash \mathsf{R}t_1t_2\cdots t_k &\Leftrightarrow (\overline{s}(t_1), \overline{s}(t_2), \ldots, \overline{s}(t_k)) \in \mathsf{R}^{\mathcal{M}} \\ &\Leftrightarrow (h(\overline{s}(t_1)), h(\overline{s}(t_2)), \ldots, h(\overline{s}(t_k))) \in \mathsf{R}^{\mathcal{N}} && \text{(since } h \text{ is a homomorphism)} \\ &\Leftrightarrow (\overline{h \circ s}(t_1), \overline{h \circ s}(t_2), \ldots, \overline{h \circ s}(t_k)) \in \mathsf{R}^{\mathcal{N}} && \text{(by part 1)} \\ &\Leftrightarrow (\mathcal{N}, h \circ s) \vDash \mathsf{R}t_1t_2\cdots t_k. \end{aligned}$$

Suppose that the statement is true for φ. We prove it for $\neg\varphi$. We have

$$\begin{aligned}(\mathcal{M}, s) \vDash \neg\varphi &\Leftrightarrow (\mathcal{M}, s) \nvDash \varphi \\ &\Leftrightarrow (\mathcal{N}, h \circ s) \nvDash \varphi && \text{(by induction)} \\ &\Leftrightarrow (\mathcal{N}, h \circ s) \vDash \neg\varphi.\end{aligned}$$

Suppose that the statement is true for φ and ψ. We have

$$\begin{aligned}(\mathcal{M}, s) \vDash \varphi \wedge \psi &\Leftrightarrow (\mathcal{M}, s) \vDash \varphi \text{ and } (\mathcal{M}, s) \vDash \psi \\ &\Leftrightarrow (\mathcal{N}, h \circ s) \vDash \varphi \text{ and } (\mathcal{N}, h \circ s) \vDash \varphi && \text{(by induction)} \\ &\Leftrightarrow (\mathcal{N}, h \circ s) \vDash \varphi \wedge \psi,\end{aligned}$$

and similarly for $\vee$ and $\rightarrow$. The result follows by induction.

4. In light of the proof of (3), we need only show that if φ is $= t_1 t_2$ where $t_1, t_2 \in Term_{\mathcal{L}}$, then $(\mathcal{M}, s) \vDash \varphi$ if and only if $(\mathcal{N}, h \circ s) \vDash \varphi$. For any $t_1, t_2 \in Term_{\mathcal{L}}$, we have

$$\begin{aligned}(\mathcal{M}, s) \vDash = t_1 t_2 &\Leftrightarrow \overline{s}(t_1) = \overline{s}(t_2) \\ &\Leftrightarrow h(\overline{s}(t_1)) = h(\overline{s}(t_2)) && \text{(since } h \text{ is injective)} \\ &\Leftrightarrow \overline{h \circ s}(t_1) = \overline{h \circ s}(t_2) && \text{(by part 1)} \\ &\Leftrightarrow (\mathcal{N}, h \circ s) \vDash = t_1 t_2.\end{aligned}$$

5. Suppose that the statement is true for φ and that $\mathsf{x} \in Var$. We have

$$\begin{aligned}(\mathcal{M}, s) \vDash \exists \mathsf{x}\varphi &\Leftrightarrow \text{There exists } a \in M \text{ such that } (\mathcal{M}, s[\mathsf{x} \Rightarrow a]) \vDash \varphi \\ &\Leftrightarrow \text{There exists } a \in M \text{ such that } (\mathcal{N}, h \circ (s[\mathsf{x} \Rightarrow a])) \vDash \varphi && \text{(by induction)} \\ &\Leftrightarrow \text{There exists } a \in M \text{ such that } (\mathcal{N}, (h \circ s)[x \Rightarrow h(a)] \vDash \varphi \\ &\Leftrightarrow \text{There exists } b \in N \text{ such that } (\mathcal{N}, (h \circ s)[x \Rightarrow b]) \vDash \varphi && \text{(since } h \text{ is bijective)} \\ &\Leftrightarrow (\mathcal{N}, h \circ s) \vDash \exists \mathsf{x}\varphi,\end{aligned}$$

and also

$$\begin{aligned}(\mathcal{M}, s) \vDash \forall \mathsf{x}\varphi &\Leftrightarrow \text{For all } a \in M, \text{ we have } (\mathcal{M}, s[\mathsf{x} \Rightarrow a]) \vDash \varphi \\ &\Leftrightarrow \text{For all } a \in M, \text{ we have } (\mathcal{N}, h \circ (s[\mathsf{x} \Rightarrow a])) \vDash \varphi && \text{(by induction)} \\ &\Leftrightarrow \text{For all } a \in M, \text{ we have } (\mathcal{N}, (h \circ s)[x \Rightarrow h(a)]) \vDash \varphi \\ &\Leftrightarrow \text{For all } b \in N, \text{ we have } (\mathcal{N}, (h \circ s)[x \Rightarrow b]) \vDash \varphi && \text{(since } h \text{ is bijective)} \\ &\Leftrightarrow (\mathcal{N}, h \circ s) \vDash \forall \mathsf{x}\varphi.\end{aligned}$$

□

Definition 4.3.7. *Let $\mathcal{M}$ and $\mathcal{N}$ be $\mathcal{L}$-structures. We say that $\mathcal{M}$ and $\mathcal{N}$ are* isomorphic *if there exists an isomorphism $h\colon \mathcal{M} \to \mathcal{N}$. In this case, we write $\mathcal{M} \cong \mathcal{N}$.*

We now introduce one of the central definitions of logic.

Definition 4.3.8. *Let $\mathcal{L}$ be a language, and let $\mathcal{M}$ and $\mathcal{N}$ be $\mathcal{L}$-structures. We write $\mathcal{M} \equiv \mathcal{N}$, and say that $\mathcal{M}$ and $\mathcal{N}$ are* elementarily equivalent*, if for all $\sigma \in Sent_{\mathcal{L}}$, we have $\mathcal{M} \vDash \sigma$ if and only if $\mathcal{N} \vDash \sigma$.*

In other words, two structures are elementarily equivalent if the same sentences are true in each structure. Notice that, by Proposition 4.2.6 we do not need to include any variable assignments in our definition, since sentences do not have free variables. The fact that a given sentence can be evaluated to a true/false value without a variable assignment is precisely what allows us to compare sentences across structures with perhaps very different underlying sets.

Corollary 4.3.9. *Let $\mathcal{L}$ be a language, and let $\mathcal{M}$ and $\mathcal{N}$ be $\mathcal{L}$-structures. If $\mathcal{M} \cong \mathcal{N}$, then $\mathcal{M} \equiv \mathcal{N}$.*

Proof. Let $\mathcal{M}$ and $\mathcal{N}$ be $\mathcal{L}$-structures, and assume that $\mathcal{M} \cong \mathcal{N}$. Fix an isomorphism $h\colon M \to N$. Let $\sigma \in Sent_{\mathcal{L}}$ be arbitrary. Fix some (any) variable assignment on M. By Theorem 4.3.6, we have

$$(\mathcal{M}, s) \vDash \sigma \text{ if and only if } (\mathcal{N}, h \circ s) \vDash \sigma.$$

Now since σ has no free variables, the variable assignments don't matter (see Notation 4.2.8), so we conclude that $\mathcal{M} \vDash \sigma$ if and only if $\mathcal{N} \vDash \sigma$. Since $\sigma \in Sent_{\mathcal{L}}$ was arbitrary, it follows that $\mathcal{M} \equiv \mathcal{N}$. □

Somewhat surprisingly, the converse of Corollary 4.3.9 is *not* true. In other words, there are elementarily equivalent structures that are not isomorphic. In fact, not only do such examples exist, but we will see that for essentially every structure $\mathcal{M}$, we can build a structure $\mathcal{N}$ with $\mathcal{M} \equiv \mathcal{N}$ but $\mathcal{M} \ncong \mathcal{N}$. Beyond how intrinsically amazing this is, it turns out to be useful. If we want to show that $\mathcal{M} \vDash \sigma$ for a given $\sigma \in Sent_{\mathcal{L}}$, we can think about going to an elementarily equivalent structure $\mathcal{N}$ that is easier to work with or understand, and show that $\mathcal{N} \vDash \sigma$. We will eventually see some extraordinary examples of arguments in this style.

Returning to substructures, notice that if $\mathcal{A}$ is a substructure of $\mathcal{M}$, then we typically have $\mathcal{A} \not\equiv \mathcal{M}$. For instance, it is possible that a subgroup of a nonabelian group is abelian. For example, if $\mathcal{M}$ is the group S_3, and $\mathcal{A}$ is the substructure of $\mathcal{M}$ with underlying set $\{id, (1\ 2)\}$ (where id is the identity function), then we have

$$\mathcal{M} \nvDash \forall \mathsf{x} \forall \mathsf{y} (\mathsf{x} * \mathsf{y} = \mathsf{y} * \mathsf{x})$$

but

$$\mathcal{A} \vDash \forall \mathsf{x} \forall \mathsf{y} (\mathsf{x} * \mathsf{y} = \mathsf{y} * \mathsf{x}).$$

Nonetheless, there are connections between when some restricted types of formulas are true in $\mathcal{A}$ versus in $\mathcal{M}$. To see this, we start with the following simple remark.

Definition 4.3.10. *Let $\mathcal{L}$ be a language, and let $QuantFreeForm_{\mathcal{L}}$ be the set of all quantifier-free formulas.*

1. *An* existential formula *is an element of $G(Sym^*_{\mathcal{L}}, QuantFreeForm_{\mathcal{L}}, \{h_{\exists, \mathsf{x}} : \mathsf{x} \in Var\})$.*
2. *A* universal formula *is an element of $G(Sym^*_{\mathcal{L}}, QuantFreeForm_{\mathcal{L}}, \{h_{\forall, \mathsf{x}} : \mathsf{x} \in Var\})$.*

In other words, an existential formula is a formula that begins with a block of existential quantifiers, and then is followed by a quantifier-free formula. A universal formula instead begins with a block of universal quantifiers, followed by a quantifier-free formula. For example,

$$\forall \mathsf{x} \forall \mathsf{y} (\mathsf{x} * \mathsf{y} = \mathsf{y} * \mathsf{x})$$

is a universal formula in the language of group theory. We can now state and prove some simple connections between how the truth of these formulas can vary between a substructure and the bigger structure.

Proposition 4.3.11. *Let $\mathcal{L}$ be a language, let $\mathcal{M}$ and $\mathcal{N}$ be $\mathcal{L}$-structures, and let $h\colon M \to N$ be an embedding. Let $s\colon Var \to M$ be a variable assignment. We have the following:*

1. *If φ is an existential formula and $(\mathcal{M}, s) \vDash \varphi$, then $(\mathcal{N}, h \circ s) \vDash \varphi$.*

2. *If φ is a universal formula, and $(\mathcal{N}, h \circ s) \vDash \varphi$, then $(\mathcal{M}, s) \vDash \varphi$*

Proof.

1. The proof is by induction on φ. For the base case, note that if φ is quantifier-free, then that statement follows from Theorem 4.3.6. Suppose that we know that the statement is true for φ, and assume that $(\mathcal{M}, s) \vDash \exists \mathsf{x} \varphi$. By definition, we can fix $a \in M$ such that $(\mathcal{M}, s[\mathsf{x} \Rightarrow a]) \vDash \varphi$. By induction, we know that $(\mathcal{N}, h \circ (s[\mathsf{x} \Rightarrow a])) \vDash \varphi$. Since $h \circ (s[\mathsf{x} \Rightarrow a]) = (h \circ s)[\mathsf{x} \Rightarrow h(a)]$, it follows that $(\mathcal{N}, (h \circ s)[\mathsf{x} \Rightarrow h(a)]) \vDash \varphi$. Therefore, $(\mathcal{N}, h \circ s) \vDash \exists \mathsf{x} \varphi$.

2. The proof is by induction on φ. For the base case, note that if φ is quantifier-free, then that statement follows from Theorem 4.3.6. Suppose that we know that the statement is true for φ, and assume that $(\mathcal{N}, h \circ s) \vDash \forall \mathsf{x} \varphi$. By definition, it follows that for every $b \in N$, we have $(\mathcal{N}, (h \circ s)[\mathsf{x} \Rightarrow b]) \vDash \varphi$.

   Now let $a \in M$ be arbitrary. Since $h(a) \in N$, it follows that $(\mathcal{N}, (h \circ s)[\mathsf{x} \Rightarrow h(a)]) \vDash \varphi$. Since $(h \circ s)[\mathsf{x} \Rightarrow h(a)] = h \circ (s[\mathsf{x} \Rightarrow a])$, we conclude that $(\mathcal{N}, h \circ (s[\mathsf{x} \Rightarrow a])) \vDash \varphi$. By induction, it follows that $(\mathcal{M}, s[\mathsf{x} \Rightarrow a]) \vDash \varphi$. Since $a \in M$ was arbitrary, we conclude that $(\mathcal{M}, s) \vDash \varphi$

□

Corollary 4.3.12. *Suppose that $\mathcal{A}$ is a substructure of $\mathcal{M}$.*

1. *For any quantifier-free formula φ and any $s \colon Var \to A$, we have*

$$(\mathcal{A}, s) \vDash \varphi \text{ if and only if } (\mathcal{M}, s) \vDash \varphi.$$

2. *For any existential formula φ and any $s \colon Var \to A$, we have*

$$\text{If } (\mathcal{A}, s) \vDash \varphi, \text{ then } (\mathcal{M}, s) \vDash \varphi.$$

3. *For any universal formula φ and any $s \colon Var \to A$, we have*

$$\text{If } (\mathcal{M}, s) \vDash \varphi, \text{ then } (\mathcal{A}, s) \vDash \varphi.$$

Proof. Immediate from Proposition 4.3.11 and Proposition 4.3.5. □

For example, the sentence σ equal to $\forall \mathsf{x} \forall \mathsf{y} (\mathsf{x} * \mathsf{y} = \mathsf{y} * \mathsf{x})$ is a universal formula in the group theory language. If $\mathcal{M}$ is a structure with $\mathcal{M} \vDash \sigma$, then whenever $\mathcal{A}$ is a substructure of $\mathcal{M}$, we have that $\mathcal{A} \vDash \sigma$. In particular, every subgroup of an abelian group is abelian. As mentioned above, the converse does not hold, and this is the reason why we only have one direction in Corollary 4.3.12 in the case of existential and universal formulas.

For more complicated formulas, we may not be able to go in either direction. For example, consider the language $\mathcal{L} = \{\mathsf{0}, \mathsf{1}, +, -, \cdot\}$ of ring theory. Let σ be the sentence

$$\forall \mathsf{x} (\neg (\mathsf{x} = \mathsf{0}) \to \exists \mathsf{y} (\mathsf{x} \cdot \mathsf{y} = \mathsf{1})),$$

which says that every nonzero element has a multiplicative inverse. Notice that σ is neither an existential formula nor a universal formula. Now if we consider the $\mathcal{L}$-structure $\mathcal{M}$ consisting of the rational field and let $\mathcal{A}$ be the substructure consisting of the integers, then $\mathcal{M} \vDash \sigma$ but $\mathcal{A} \nvDash \sigma$. In contrast, if we consider the $\mathcal{L}$-structure $\mathcal{M}$ consisting of the polynomial ring $\mathbb{R}[x]$ and let $\mathcal{A}$ be the substructure consisting of the real numbers (i.e. constant polynomials), then $\mathcal{M} \nvDash \sigma$ but $\mathcal{A} \vDash \sigma$. In other words, if a formula is neither existential nor universal, it is possible that the truth of the formula in a structure has no relation to its truth in a substructure.

We end this section with another simple way to take a structure and turn it into a "smaller" one. In this scenario, we simply drop some symbols from our language.

Definition 4.3.13. *Let $\mathcal{L}' \subseteq \mathcal{L}$ be languages, let $\mathcal{M}'$ be an $\mathcal{L}'$-structure, and let $\mathcal{M}$ be an $\mathcal{L}$-structure. We say that $\mathcal{M}'$ is the* restriction *(or* reduct*) of $\mathcal{M}$ to $\mathcal{L}$, and that $\mathcal{M}$ is an* expansion *of $\mathcal{M}'$ to $\mathcal{L}$, if the following conditions hold:*

- $M = M'$.
- $\mathsf{c}^{\mathcal{M}'} = \mathsf{c}^{\mathcal{M}}$ *for all* $\mathsf{c} \in \mathcal{C}'$.
- $\mathsf{R}^{\mathcal{M}'} = \mathsf{R}^{\mathcal{M}}$ *for all* $\mathsf{R} \in \mathcal{R}'$.
- $\mathsf{f}^{\mathcal{M}'} = \mathsf{f}^{\mathcal{M}}$ *for all* $\mathsf{f} \in \mathcal{F}'$.

In this case, we write $\mathcal{M}' = \mathcal{M} \upharpoonright \mathcal{L}$.

For example, suppose that $\mathcal{L} = \{\mathsf{0}, \mathsf{1}, +, -, \cdot\}$ is the language of rings and that $\mathcal{L}' = \{\mathsf{0}, +, -\}$ is the language of groups (we have changed the symbols in the latter from $\mathsf{0}, +, -\mathsf{1}$, but only the types and arities of the symbols really matter). Given an $\mathcal{L}$ structure $\mathcal{M}$, we can form the restriction $\mathcal{M}'$ of $\mathcal{M}$ to $\mathcal{L}'$ by keeping the same underlying set and same interpretations of $\mathsf{0}$, $+$, and $-$, but by also "forgetting" about $\mathsf{1}$ and $\cdot$. Notice that if $\mathcal{M}$ happened to be a ring, then $\mathcal{M}'$ will be an abelian group.

In contrast to substructures, it is intuitively clear that truth is preserved between a structure and its restriction. Here is the formal statement.

Proposition 4.3.14. *Let $\mathcal{L}' \subseteq \mathcal{L}$ be languages, let $\mathcal{M}$ be an $\mathcal{L}$-structure, and let $\mathcal{M}' = \mathcal{M} \upharpoonright \mathcal{L}$ be the restriction of $\mathcal{M}$ to $\mathcal{L}'$. For all $\varphi \in Form_{\mathcal{L}'}$ and all $s\colon Var \to M$, we have $(\mathcal{M}, s) \vDash \varphi$ if and only if $(\mathcal{M}', s) \vDash \varphi$.*

Proof. By a trivial induction. □

4.4 Definability

A wonderful side-effect of developing a formal language is the ability to talk about what objects we can define using that language.

Definition 4.4.1. *Let $\mathcal{M}$ be an $\mathcal{L}$-structure, let $k \in \mathbb{N}^+$, and let $X \subseteq M^k$. We say that X is* definable *in $\mathcal{M}$ if there exists $\varphi(\mathsf{x}_1, \mathsf{x}_2, \ldots, \mathsf{x}_k) \in Form_{\mathcal{L}}$ such that*

$$X = \{(a_1, a_2, \ldots, a_k) \in M^k : (\mathcal{M}, a_1, a_2, \ldots, a_k) \vDash \varphi\}.$$

In other words, a set $X \subseteq M^k$ is definable in $\mathcal{M}$ if we can find a formula φ with k free variables such that the formula is true in $\mathcal{M}$ when we interpret the free variables as a k-tuple in X, and is false otherwise. For example, let $\mathcal{L} = \{\mathsf{0}, \mathsf{1}, +, \cdot\}$, where $\mathsf{0}$ and $\mathsf{1}$ are constant symbols and $+$ and $\cdot$ are binary function symbols.

1. The set $X = \{(m, n) \in \mathbb{N}^2 : m < n\}$ is definable in the structure $(\mathbb{N}, 0, 1, +, \cdot)$ as witnessed by the formula
$$\exists \mathsf{z}(\mathsf{z} \neq \mathsf{0} \wedge (\mathsf{x} + \mathsf{z} = \mathsf{y})).$$
2. The set $X = \{n \in \mathbb{N} : n \text{ is prime}\}$ is definable in the structure $(\mathbb{N}, 0, 1, +, \cdot)$ as witnessed by the formula
$$\neg(\mathsf{x} = \mathsf{1}) \wedge \forall \mathsf{y} \forall \mathsf{z}(\mathsf{x} = \mathsf{y} \cdot \mathsf{z} \to (\mathsf{y} = \mathsf{1} \vee \mathsf{z} = \mathsf{1})).$$
3. The set $X = \{r \in \mathbb{R} : r \geq 0\}$ is definable in the structure $(\mathbb{R}, 0, 1, +, \cdot)$ as witnessed by the formula
$$\exists \mathsf{y}(\mathsf{y} \cdot \mathsf{y} = \mathsf{x}).$$

Let's consider another language. Let $\mathcal{L} = \{<\}$ where $<$ is a binary relation symbol. For every $n \in \mathbb{N}$, the set $\{n\}$ is definable in $(\mathbb{N}, <)$. To see this, for each $n\mathbb{N}^+$, let $\varphi_n(\mathsf{x})$ be the formula

$$\exists \mathsf{y}_1 \exists \mathsf{y}_2 \cdots \exists \mathsf{y}_\mathsf{n} (\bigwedge_{1 \leq \mathsf{i} < \mathsf{j} \leq \mathsf{n}} (\mathsf{y}_\mathsf{i} \neq \mathsf{y}_\mathsf{j}) \wedge \bigwedge_{\mathsf{i}=1}^{\mathsf{n}} (\mathsf{y}_\mathsf{i} < \mathsf{x})).$$

For each $n \in \mathbb{N}^+$, the formula $\varphi_n(\mathsf{x})$ defines the set $\{k \in \mathbb{N} : n \leq k\}$. Now notice that $\{0\}$ is definable as witnessed by the formula

$$\neg \exists \mathsf{y}(\mathsf{y} < \mathsf{x}),$$

and for each $n \in \mathbb{N}^+$, the set $\{n\}$ is definable as witnessed by the formula

$$\varphi_n(\mathsf{x}) \wedge \neg \varphi_{n+1}(\mathsf{x}).$$

Finally, let $\mathcal{L} = \{\mathsf{e}, \mathsf{f}\}$ be the restricted group theory language. Let $(G, e, \cdot)$ be a group interpreted as an $\mathcal{L}$-structure. The center of G is definable in $(G, e, \cdot)$ as witnessed by the formula

$$\forall \mathsf{y}(\mathsf{f}(\mathsf{x}, \mathsf{y}) = \mathsf{f}(\mathsf{y}, \mathsf{x})).$$

Sometimes, there isn't an obvious way to show that a set is definable, but some cleverness and/or nontrivial mathematics comes to the rescue. In each of the examples below, let $\mathcal{L} = \{\mathsf{0}, \mathsf{1}, +, \cdot\}$ be the language described above.

1. The set $\mathbb{N}$ is definable in $(\mathbb{Z}, 0, 1, +, \cdot)$ as witnessed by the formula

   $$\exists \mathsf{y}_1 \exists \mathsf{y}_2 \exists \mathsf{y}_3 \exists \mathsf{y}_4 (\mathsf{x} = \mathsf{y}_1 \cdot \mathsf{y}_1 + \mathsf{y}_2 \cdot \mathsf{y}_2 + \mathsf{y}_3 \cdot \mathsf{y}_3 + \mathsf{y}_4 \cdot \mathsf{y}_4).$$

   Certainly every element of $\mathbb{Z}$ that is a sum of squares must be an element of $\mathbb{N}$. The fact that every element of $\mathbb{N}$ is a sum of four squares is Lagrange's Theorem, an important result in number theory.

2. Let $(R, 0, 1, +, \cdot)$ be a commutative ring. The Jacobson radical of R, denoted $Jac(R)$ is the intersection of all maximal ideals of R. As stated, it is not clear that this is definable in $(R, 0, 1, +, \cdot)$ because it appears to quantify over subsets. However, a basic result in commutative algebra says that

   $$\begin{aligned} a \in Jac(R) &\Leftrightarrow ab - 1 \text{ is a unit for all } b \in R \\ &\Leftrightarrow \text{For all } b \in R, \text{ there exists } c \in R \text{ with } (ab-1)c = 1. \end{aligned}$$

   Using this, it follows that $Jac(R)$ is definable in $(R, 0, 1, +, \cdot)$ as witnessed by the formula

   $$\forall \mathsf{y} \exists \mathsf{z}((\mathsf{x} \cdot \mathsf{y}) \cdot \mathsf{z} = \mathsf{z} + 1).$$

3. The set $\mathbb{Z}$ is definable in $(\mathbb{Q}, 0, 1, +, \cdot)$. This is a deep result of Julia Robinson using some nontrivial number theory.

4. The set $X = \{(k, m, n) \in \mathbb{N}^3 : k^m = n\}$ is definable in $(\mathbb{N}, 0, 1, +, \cdot)$, as is the set

   $$\{(m, n) \in \mathbb{N}^2 : m \text{ is the } n^{\text{th}} \text{ digit in the decimal expansion of } \pi\}.$$

   In fact, every set $C \subseteq \mathbb{N}^k$ which is "computable" (i.e. for which it is possible to write a computer program that outputs *yes* on elements of C and *no* on elements of $\mathbb{N}^k \backslash C$) is definable in $(\mathbb{N}, 0, 1, +, \cdot)$. We will prove this challenging, but fundamental, result later (see Corollary 14.1.5).

The collection of definable sets in a structure have some simple closure properties.

Proposition 4.4.2. *Let $\mathcal{M}$ be an $\mathcal{L}$-structure, and let $k \in \mathbb{N}^+$. Let*

$$\mathcal{D}_k = \{X \in \mathcal{P}(M^k) : X \text{ is definable in } \mathcal{M}\}.$$

We have the following:

1. *If $X, Y \in \mathcal{D}_k$, then $X \cup Y \in \mathcal{D}_k$.*
2. *If $X, Y \in \mathcal{D}_k$, then $X \cap Y \in \mathcal{D}_k$.*
3. *If $X \in \mathcal{D}_k$, then $M \backslash X \in \mathcal{D}_k$.*

Proof. Each of these follow by taking the $\vee$, $\wedge$, and $\neg$ of the respective formulas. □

Determining which sets in a structure are definable is typically a challenging task, but classifying the collection of definable sets is one of the primary goals when seeking to fully "understand" a structure. Let's return to the language $\mathcal{L} = \{\mathsf{0}, \mathsf{1}, +, \cdot\}$. Here we outline several important restrictions on definable sets in a few natural $\mathcal{L}$-structures. We will prove each of these facts later.

1. In the structure $(\mathbb{C}, 0, 1, +, \cdot)$, every subset of $\mathbb{C}$ that is definable is either finite or cofinite (i.e. its complement is finite). The converse is not true. For example, any element of $\mathbb{C}$ that is transcendental over $\mathbb{Q}$ is not an element of any finite definable set.
2. In the structure $(\mathbb{R}, 0, 1, +, \cdot)$, every subset of $\mathbb{R}$ that is definable is a finite union of intervals and points (but again, the converse is not true).
3. In the structure $(\mathbb{N}, 0, 1, +, \cdot)$, every computable subset of $\mathbb{N}$ is definable. In fact, many other subsets of $\mathbb{Z}$ are also definable, and it is challenging to describe a subset of $\mathbb{N}$ that is not definable (although we will eventually give an example in Theorem 14.2.2).
4. Since $\mathbb{N}$ is definable in $(\mathbb{Z}, 0, 1, +, \cdot)$, it turns out that the definable subsets of $\mathbb{Z}$ are also rich and complicated.
5. Since $\mathbb{Z}$ is definable in $(\mathbb{Q}, 0, 1, +, \cdot)$, the definable subsets of $\mathbb{Q}$ are similarly complicated.

Just like for elementary classes, it is clear how to attempt to show that something is definable (although as we've seen this may require a great deal of cleverness). However, it is not at all obvious how one could show that a set is not definable. Fortunately, Theorem 4.3.6 can be used to prove negative results about definability.

Definition 4.4.3. *Let $\mathcal{M}$ be an $\mathcal{L}$-structure. An isomorphism $h\colon M \to M$ is called an* automorphism.

Proposition 4.4.4. *Suppose that $\mathcal{M}$ is an $\mathcal{L}$-structure and $k \in \mathbb{N}^+$. Suppose also that $X \subseteq M^k$ is definable in $\mathcal{M}$ and that $h\colon M \to M$ is an automorphism. For every $a_1, a_2, \ldots, a_k \in M$, we have*

$$(a_1, a_2, \ldots, a_k) \in X \text{ if and only if } (h(a_1), h(a_2), \ldots, h(a_k)) \in X.$$

Proof. Fix $\varphi(\mathsf{x}_1, \mathsf{x}_2, \ldots, \mathsf{x}_k) \in Form_{\mathcal{L}}$ such that

$$X = \{(a_1, a_2, \ldots, a_k) \in M^k : (\mathcal{M}, a_1, a_2, \ldots, a_k) \vDash \varphi\}.$$

By part 4 of Theorem 4.3.6, we know that for every $a_1, a_2, \ldots, a_k \in M$, we have

$$(\mathcal{M}, a_1, a_2, \ldots, a_k) \vDash \varphi \text{ if and only if } (\mathcal{M}, h(a_1), h(a_2), \ldots, h(a_k)) \vDash \varphi.$$

Therefore, for every $a_1, a_2, \ldots, a_k \in M$, we have

$$(a_1, a_2, \ldots, a_k) \in X \text{ if and only if } (h(a_1), h(a_2), \ldots, h(a_k)) \in X.$$

□

In other words, automorphisms of a structure must fix definable sets as a whole (note that this is *not* saying that an automorphism must fix definable sets pointwise). Therefore, in order to show that a given set is not definable, we can find an automorphism that does not fix the set.

Corollary 4.4.5. *Suppose that $\mathcal{M}$ is an $\mathcal{L}$-structure and $k \in \mathbb{N}^+$. Suppose also that $X \subseteq M^k$ and that $h\colon M \to M$ is an automorphism. Suppose that there exists $a_1, a_2, \ldots, a_k \in M$ such that exactly one of the following holds:*

- $(a_1, a_2, \ldots, a_k) \in X$.
- $(h(a_1), h(a_2), \ldots, h(a_k)) \in X$.

We then have that X is not definable in $\mathcal{M}$.

For example, let $\mathcal{L} = \{\mathsf{R}\}$ where R is a binary relation symbol, and let $\mathcal{M}$ be the $\mathcal{L}$-structure where $M = \mathbb{Z}$ and $\mathsf{R}^{\mathcal{M}} = \{(a, b) \in \mathbb{Z}^2 : a < b\}$. We show that a set $X \subseteq M$ is definable in $\mathcal{M}$ if and only if either $X = \emptyset$ or $X = \mathbb{Z}$. First notice that $\emptyset$ is definable as witnessed by $\neg(\mathsf{x} = \mathsf{x})$ and $\mathbb{Z}$ as witnessed by $\mathsf{x} = \mathsf{x}$. Suppose now that $X \subseteq \mathbb{Z}$ is such that $X \neq \emptyset$ and $X \neq \mathbb{Z}$. Fix $a, b \in \mathbb{Z}$ such that $a \in X$ and $b \notin X$. Define $h\colon M \to M$ by letting $h(c) = c + (b - a)$ for all $c \in M$. Notice that h is automorphism of $\mathcal{M}$ because it is bijective (the map $g(c) = c - (b - a)$ is clearly an inverse) and a homomorphism. For the latter, notice that if $c_1, c_2 \in \mathbb{Z}$ are arbitrary, then have have

$$\begin{aligned}(c_1, c_2) \in \mathsf{R}^{\mathcal{M}} &\Leftrightarrow c_1 < c_2 \\ &\Leftrightarrow c_1 + (b - a) < c_2 + (b - a) \\ &\Leftrightarrow h(c_1) < h(c_2) \\ &\Leftrightarrow (h(c_1), h(c_2)) \in \mathsf{R}^{\mathcal{M}}.\end{aligned}$$

Notice also that $h(a) = a + (b - a) = b$, so $a \in X$ but $h(a) \notin X$. Using Corollary 4.4.5, it follows that X is not definable in $\mathcal{M}$.

Definition 4.4.6. *Let $\mathcal{M}$ be an $\mathcal{L}$-structure. Suppose that $k \in \mathbb{N}^+$ and $X \subseteq M^k$. We say that X is* definable with parameters *in $\mathcal{M}$ if there exists $\varphi(\mathsf{x}_1, \mathsf{x}_2, \ldots, \mathsf{x}_k, \mathsf{y}_1, \mathsf{y}_2, \ldots, \mathsf{y}_n) \in Form_{\mathcal{L}}$ together with $b_1, b_2, \ldots, b_n \in M$ such that*

$$X = \{(a_1, a_2, \ldots, a_k) \in M^k : (\mathcal{M}, a_1, a_2, \ldots, a_k, b_1, b_2, \ldots, b_n) \vDash \varphi\}$$

Intuitively, a set $X \subseteq M^k$ is definable with parameters if we are allowed to use specific elements of M inside our formulas to help us define it. For example, for each $n \in \mathbb{Z}$ the set $\{n\}$ is trivially definable with parameters in the structure $\mathcal{M} = (\mathbb{Z}, <)$: simply let $\varphi(\mathsf{x}, \mathsf{y})$ be the formula $\mathsf{x} = \mathsf{y}$, and notice that for each $a \in \mathbb{Z}$, we have

$$(\mathcal{M}, a, n) \vDash \varphi \Leftrightarrow a = n.$$

In fact, given *any* structure $\mathcal{M}$, the set $\{m\}$ is definable with parameters in $\mathcal{M}$.

Let's return to the language $\mathcal{L} = \{0, 1, +, \cdot\}$. Above, we talked about restrictions on definable sets in several natural $\mathcal{L}$-structures. We now discuss the situation when we switch to looking at those sets that are definable with parameters.

1. In the structure $(\mathbb{C}, 0, 1, +, \cdot)$, a subset of $\mathbb{C}$ is definable with parameters if and only if it is either finite or cofinite.
2. In the structure $(\mathbb{R}, 0, 1, +, \cdot)$, a subset of $\mathbb{R}$ is definable with parameters if and only if it is finite union of intervals and points.
3. In the structures with underlying sets $\mathbb{N}$, $\mathbb{Z}$, and $\mathbb{Q}$, we can already define each specific element without parameters, so it turns out that a set is definable with parameters if and only if it is definable.

We will come back to discuss definability with parameters in Section 7.1.

4.5 Elementary Substructures and Elementary Maps

Suppose that $\mathcal{A}$ is a substructure of $\mathcal{M}$. In Corollary 4.3.12, we showed that for all quantifier-free formulas φ and all $s\colon Var \to A$, we have

$$(\mathcal{A}, s) \vDash \varphi \text{ if and only if } (\mathcal{M}, s) \vDash \varphi.$$

Once we look at more general formulas involving quantifiers, this connection can easily break down. After all, a quantifier ranges over the entire underlying set, so if A is a proper subset of M, the recursive definitions of $\vDash$ will greatly differ. Regardless, it is at least conceivable that a proper substructure of a larger structure might occasionally satisfy the same formulas. We give these (at the moment seemingly magical) substructures a special name.

Definition 4.5.1. *Let $\mathcal{L}$ be a language, let $\mathcal{M}$ be an $\mathcal{L}$-structure, and let $\mathcal{A}$ be a substructure of $\mathcal{M}$. We say that $\mathcal{A}$ is an* elementary substructure *if for all $\varphi \in Form_{\mathcal{L}}$ and all $s\colon Var \to A$, we have*

$$(\mathcal{A}, s) \vDash \varphi \text{ if and only if } (\mathcal{M}, s) \vDash \varphi.$$

We write $\mathcal{A} \preceq \mathcal{M}$ to mean that $\mathcal{A}$ is an elementary substructure of $\mathcal{M}$.

In our other notation, a substructure $\mathcal{A}$ of $\mathcal{M}$ is an elementary substructure if for all $\varphi(\mathsf{x}_1, \mathsf{x}_2, \ldots, \mathsf{x}_n) \in Form_{\mathcal{L}}$ and all $a_1, a_2, \ldots, a_n \in A$, we have

$$(\mathcal{A}, a_1, a_2, \ldots, a_n) \vDash \varphi \text{ if and only if } (\mathcal{M}, a_1, a_2, \ldots, a_n) \vDash \varphi.$$

Notice that if $\mathcal{A}$ is an elementary substructure of $\mathcal{M}$, then in particular we have that $\mathcal{A} \equiv \mathcal{M}$, i.e. that $\mathcal{A}$ is elementarily equivalent to $\mathcal{M}$. To see this, simply notice that every $\sigma \in Sent_{\mathcal{L}}$ is in particular a formula, and that variable assignments do not affect the truth of sentences (since sentences have no free variables).

However, it is possible to have a substructure $\mathcal{A}$ of $\mathcal{M}$ with $\mathcal{A} \equiv \mathcal{M}$, but where $\mathcal{A} \not\preceq \mathcal{M}$. For example, let $\mathcal{L} = \{\mathsf{f}\}$ where f is a unary function symbol. Let $\mathcal{M}$ be the $\mathcal{L}$-structure with $M = \mathbb{N}$ and $\mathsf{f}^{\mathcal{M}}(n) = n + 1$. Let $\mathcal{A}$ be $\mathcal{L}$-structure with $A = \mathbb{N}^+$ and $\mathsf{f}^{\mathcal{A}}(n) = n + 1$. We then have that $\mathcal{A}$ is a substructure of $\mathcal{M}$. Now $\mathcal{A} \cong \mathcal{M}$ via the function $h(m) = m - 1$, so $\mathcal{A} \equiv \mathcal{M}$ by Corollary 4.3.9. However, notice that $\mathcal{A} \not\preceq \mathcal{M}$ because if $\varphi(\mathsf{x})$ is the formula $\exists \mathsf{y}(\mathsf{fy} = \mathsf{x})$, we then have that $(\mathcal{A}, 1) \nvDash \varphi$ but $(\mathcal{M}, 1) \vDash \varphi$. Generalizing this example, one can show that the only elementary substructure of $\mathcal{M}$ is the structure $\mathcal{M}$ itself (noting that $\mathcal{A}$ has to have a smallest element by well-ordering, then using the above argument to argue that this smallest element is 0, and finally using the fact that A must be closed under the successor function).

Before jumping into a result that will greatly simplify the construction of elementary substructures, we start with a simple lemma that will allow us to turn one type of quantifier into the other.

Lemma 4.5.2. *Let $\mathcal{M}$ be an $\mathcal{L}$-structure, and let $s\colon Var \to M$ be a variable assignment. For any $\varphi \in Form_{\mathcal{L}}$, we have*

$$(\mathcal{M}, s) \vDash \forall x \varphi \text{ if and only if } (\mathcal{M}, s) \vDash \neg \exists x \neg \varphi.$$

Proof. Let $\varphi \in Form_{\mathcal{L}}$ be arbitrary. Using the recursive definition of $\vDash$, we have

$$\begin{aligned}
(\mathcal{M}, s) \vDash \forall x \varphi &\Leftrightarrow \text{For all } a \in M, \text{ we have } (\mathcal{M}, s[\mathsf{x} \Rightarrow a]) \vDash \varphi \\
&\Leftrightarrow \text{For all } a \in M, \text{ we have } (\mathcal{M}, s[\mathsf{x} \Rightarrow a]) \nvDash \neg \varphi \\
&\Leftrightarrow \text{There does not exist } a \in M \text{ with } (\mathcal{M}, s[\mathsf{x} \Rightarrow a]) \vDash \neg \varphi \\
&\Leftrightarrow (\mathcal{M}, s) \nvDash \exists \mathsf{x} \neg \varphi \\
&\Leftrightarrow (\mathcal{M}, s) \vDash \neg \exists \mathsf{x} \neg \varphi.
\end{aligned}$$

□

As mentioned above, the quantifiers are the obstacle in pushing Corollary 4.3.12 from simple formulas to more complex ones. The next test simplifies the process to that of checking one existential quantifier at a time. Notice that the condition given only refers to truth in the structure $\mathcal{M}$, and hence does not reference truth in $\mathcal{A}$.

Theorem 4.5.3 (Tarski-Vaught Test)**.** *Suppose that $\mathcal{A}$ is a substructure of $\mathcal{M}$. The following are equivalent:*

1. *$\mathcal{A} \preceq \mathcal{M}$, i.e. $\mathcal{A}$ is an elementary substructure of $\mathcal{M}$.*
2. *Whenever $\varphi \in Form_{\mathcal{L}}$, $\mathsf{x} \in Var$, and $s\colon Var \to A$ satisfy $(\mathcal{M}, s) \vDash \exists \mathsf{x}\varphi$, there exists $a \in A$ such that*

$$(\mathcal{M}, s[\mathsf{x} \Rightarrow a]) \vDash \varphi.$$

Proof. We first prove that 1 implies 2. Suppose then that $\mathcal{A} \preceq \mathcal{M}$. Let $\varphi \in Form_{\mathcal{L}}$ and $s\colon Var \to A$ be arbitrary such that $(\mathcal{M}, s) \vDash \exists \mathsf{x}\varphi$. Using the fact that $\mathcal{A} \preceq \mathcal{M}$, it follows that $(\mathcal{A}, s) \vDash \exists \mathsf{x}\varphi$. Fix $a \in A$ such that $(\mathcal{A}, s[\mathsf{x} \Rightarrow a]) \vDash \varphi$. Using again the fact that $\mathcal{A} \preceq \mathcal{M}$, we have $(\mathcal{M}, s[\mathsf{x} \Rightarrow a]) \vDash \varphi$.

We now prove that 2 implies 1. We prove by induction on $\varphi \in Form_{\mathcal{L}}$ that for all $s\colon Var \to A$, we have $(\mathcal{A}, s) \vDash \varphi$ if and only if $(\mathcal{M}, s) \vDash \varphi$. That is, we let

$$X = \{\varphi \in Form_{\mathcal{L}} : \text{ For all } s\colon Var \to A \text{ we have } (\mathcal{A}, s) \vDash \varphi \text{ if and only if } (\mathcal{M}, s) \vDash \varphi\},$$

and prove that $X = Form_{\mathcal{L}}$ by induction. First notice that $\varphi \in X$ for all quantifier-free φ by Corollary 4.3.12.

Suppose now that $\varphi \in X$. For any $s\colon Var \to A$, we have

$$\begin{aligned} (\mathcal{A}, s) \vDash \neg\varphi &\Leftrightarrow (\mathcal{A}, s) \nvDash \varphi \\ &\Leftrightarrow (\mathcal{M}, s) \nvDash \varphi && \text{(since } \varphi \in X\text{)} \\ &\Leftrightarrow (\mathcal{M}, s) \vDash \neg\varphi \end{aligned}$$

Therefore, $\neg\varphi \in X$.

Suppose now that $\varphi, \psi \in X$. For any $s\colon Var \to A$, we have

$$\begin{aligned} (\mathcal{A}, s) \vDash \varphi \wedge \psi &\Leftrightarrow (\mathcal{A}, s) \vDash \varphi \text{ and } (\mathcal{A}, s) \vDash \psi \\ &\Leftrightarrow (\mathcal{M}, s) \vDash \varphi \text{ and } (\mathcal{M}, s) \vDash \psi && \text{(since } \varphi, \psi \in X\text{)} \\ &\Leftrightarrow (\mathcal{M}, s) \vDash \varphi \wedge \psi \end{aligned}$$

Therefore, $\varphi \wedge \psi \in X$. Similarly, we have $\varphi \vee \psi \in X$ and $\varphi \to \psi \in X$.

Suppose now that $\varphi \in X$ and $\mathsf{x} \in Var$. For any $s\colon Var \to A$, we have

$$\begin{aligned} (\mathcal{A}, s) \vDash \exists \mathsf{x}\varphi &\Leftrightarrow \text{There exists } a \in A \text{ such that } (\mathcal{A}, s[\mathsf{x} \Rightarrow a]) \vDash \varphi \\ &\Leftrightarrow \text{There exists } a \in A \text{ such that } (\mathcal{M}, s[\mathsf{x} \Rightarrow a]) \vDash \varphi && \text{(since } \varphi \in X\text{)} \\ &\Leftrightarrow (\mathcal{M}, s) \vDash \exists \mathsf{x}\varphi && \text{(by our assumption 2).} \end{aligned}$$

Therefore, $\exists \mathsf{x}\varphi \in X$.

Suppose now that $\varphi \in X$ and $\mathsf{x} \in Var$. We then have that $\neg\varphi \in X$ from above, hence $\exists \mathsf{x}\neg\varphi \in X$ from above, hence $\neg\exists \mathsf{x}\neg\varphi \in X$ again from above. Thus, for any $s\colon Var \to A$, we have

$$\begin{aligned} (\mathcal{A}, s) \vDash \forall \mathsf{x}\varphi &\Leftrightarrow (\mathcal{A}, s) \vDash \neg\exists \mathsf{x}\neg\varphi && \text{(by Lemma 4.5.2)} \\ &\Leftrightarrow (\mathcal{M}, s) \vDash \neg\exists \mathsf{x}\neg\varphi && \text{(since } \neg\exists \mathsf{x}\neg\varphi \in X\text{)} \\ &\Leftrightarrow (\mathcal{M}, s) \vDash \forall \mathsf{x}\varphi && \text{(by Lemma 4.5.2).} \end{aligned}$$

Therefore, $\forall \mathsf{x}\varphi \in X$. □

The Tarski-Vaught Test gives an interesting way to build an elementary substructure of a given structure $\mathcal{M}$. Start by taking the set $A_0 = \{\mathsf{c}^{\mathcal{M}} : \mathsf{c} \in \mathcal{C}\}$, which must be a subset of the universe of any substructure. Now we need to do two things. First, by Proposition 4.3.2, we need to ensure that our set is closed under the functions $\mathsf{f}^{\mathcal{M}}$. Suppose that we close off A_0 under all of the functions and end up with a set A_1. Now we need to make sure that the Tarski-Vaught criterion is satisfied. The idea is to look at a formula $\varphi(\mathsf{y}_1, \mathsf{y}_2, \ldots, \mathsf{y}_k, \mathsf{x}) \in Form_{\mathcal{L}}$ together with an assignment sending each y_i to an $a_i \in A_1$. If we find that

$$(\mathcal{M}, a_1, a_2, \ldots, a_k) \vDash \exists \mathsf{x} \varphi(\mathsf{y}_1, \mathsf{y}_2, \ldots, \mathsf{y}_k, \mathsf{x}),$$

then we can fix an $m \in M$ with

$$(\mathcal{M}, a_1, a_2, \ldots, a_k, m) \vDash \varphi(\mathsf{y}_1, \mathsf{y}_2, \ldots, \mathsf{y}_k, \mathsf{x}).$$

The idea then is to add m to our set A_1, so that our substructure will now have a witness to this existential statement. In fact, we want to fix a witnessing m for *each* formula and assignment of free variables to A_1, and add all of the resulting m to A_1 in order form another set A_2. Now A_2 may not be closed under the functions $\mathsf{f}^{\mathcal{M}}$. Moreover, if we allow variable assignments that take values in A_2 (rather than just A_1), then we may now have even *more* existential witnesses that we need to consider. The idea is to keep iterating the process of closing off under the functions $\mathsf{f}^{\mathcal{M}}$ and adding existential witnesses. In other words, we want to generate a set using these processes.

We formalize these ideas in the following fundamental theorem. We even generalize it a bit by allowing us to start with any countable set X that we want to include in our elementary substructure.

Theorem 4.5.4 (Countable Lowenheim-Skolem-Tarski Theorem)**.** *Suppose that $\mathcal{L}$ is countable (i.e. each of the set $\mathcal{C}$, $\mathcal{R}$, and $\mathcal{F}$ are countable), that $\mathcal{M}$ is an $\mathcal{L}$-structure, and that $X \subseteq M$ is countable. There exists a countable $\mathcal{A} \preceq \mathcal{M}$ such that $X \subseteq A$.*

Proof. Since structures are nonempty, we first fix an element $d \in M$. We will use d as "dummy" element of M to ensure that we always have something to go to when all else fails.

- For each $\varphi \in Form_{\mathcal{L}}$ and $\mathsf{x} \in Var$ such that $FreeVar(\varphi) = \{\mathsf{x}\}$, we define an element $n_{\varphi,\mathsf{x}} \in M$ as follows. If $\mathcal{M} \vDash \exists \mathsf{x} \varphi$, fix an arbitrary $m \in M$ such that $(\mathcal{M}, m) \vDash \varphi$, and let $n_{\varphi,\mathsf{x}} = m$. Otherwise, let $n_{\varphi,\mathsf{x}} = d$.

- Now for each $\varphi \in Form_{\mathcal{L}}$ and $\mathsf{x} \in Var$ such that $\{\mathsf{x}\} \subsetneq FreeVar(\varphi)$, we define a function. Suppose that $FreeVar(\varphi) = \{\mathsf{y}_1, \mathsf{y}_2, \ldots, \mathsf{y}_k, \mathsf{x}\}$. We define a function $h_{\varphi,\mathsf{x}} \colon M^k \to M$ as follows. Let $a_1, a_2, \ldots, a_k \in M$ be arbitrary. If $(\mathcal{M}, a_1, a_2, \ldots, a_k) \vDash \exists \mathsf{x} \varphi$, fix some $b \in M$ such that $(\mathcal{M}, a_1, a_2, \ldots, a_k, b) \vDash \varphi$, and let $h_{\varphi,\mathsf{x}}(a_1, a_2, \ldots, a_k) = b$. Otherwise, let $h_{\varphi,\mathsf{x}}(a_1, a_2, \ldots, a_k) = d$.

We now start with the set

$$B = X \cup \{d\} \cup \{\mathsf{c}^{\mathcal{M}} : \mathsf{c} \in \mathcal{C}\} \cup \{n_{\varphi,\mathsf{x}} : \varphi \in Form_{\mathcal{L}}, \mathsf{x} \in Var, \text{ and } FreeVar(\varphi) = \{\mathsf{x}\}\},$$

and generate by closing off under the functions $f^{\mathcal{M}}$ and $h_{\varphi,\mathsf{x}}$. In other words, we let

$$A = G(M, B, \{\mathsf{f}^{\mathcal{M}} : \mathsf{f} \in \mathcal{F}_k\} \cup \{h_{\varphi,\mathsf{x}} : \varphi \in Form_P, \mathsf{x} \in FreeVar(\varphi), \text{ and } |FreeVar(\varphi)| \geq 2\}).$$

Since $\mathcal{L}$ is countable, we have that $Form_{\mathcal{L}}$ is countable by Exercise 6 in Chapter 2. Since Var is also countable, it follows that $Form_{\mathcal{L}} \times Var$ is countable. Combining this with the fact that both X and $\mathcal{C}$ are countable, it follows that B is countable. Moreover, using the fact that $\mathcal{F}$ is countable, it follows that

$$\{\mathsf{f}^{\mathcal{M}} : \mathsf{f} \in \mathcal{F}_k\} \cup \{h_{\varphi,\mathsf{x}} : \varphi \in Form_P, \mathsf{x} \in FreeVar(\varphi), \text{ and } |FreeVar(\varphi)| \geq 2\}$$

is countable. Using Exercise 6 in Chapter 2 again, we conclude that A is countable. Since A is closed under the functions $\mathsf{f}^{\mathcal{M}}$, Proposition 4.3.2 implies that A is the universe of a substructure $\mathcal{A}$ of $\mathcal{M}$. Notice also that $X \subseteq A$ since $X \subseteq B$.

Thus, we need only show that $\mathcal{A} \preceq \mathcal{M}$, which we do by using the Tarski-Vaught test. Let $\varphi \in Form_{\mathcal{L}}$, $\mathsf{x} \in Var$, and $s\colon Var \to A$ be arbitrary such that $(\mathcal{M}, s) \vDash \exists \mathsf{x}\varphi$.

- Suppose first that $\mathsf{x} \notin FreeVar(\varphi)$. Since $(\mathcal{M}, s) \vDash \exists \mathsf{x}\varphi$, we may fix $m \in \mathcal{M}$ such that $(\mathcal{M}, s[\mathsf{x} \Rightarrow m]) \vDash \varphi$. Now using the fact that $\mathsf{x} \notin FreeVar(\varphi)$, it follows that $(\mathcal{M}, s[\mathsf{x} \Rightarrow d]) \vDash \varphi$.
- Suppose now that $FreeVar(\varphi) = \{\mathsf{x}\}$, and let $a = n_{\varphi,\mathsf{x}} \in A$. Since $\mathcal{M} \vDash \exists \mathsf{x}\varphi$, we have $(\mathcal{M}, a) \vDash \varphi$ by definition of $n_{\varphi,\mathsf{x}}$, so there exists $a \in A$ such that $(\mathcal{M}, s[\mathsf{x} \Rightarrow a]) \vDash \varphi$.
- Finally, suppose that $FreeVar(\varphi) = \{\mathsf{y}_1, \mathsf{y}_2, \ldots, \mathsf{y}_k, \mathsf{x}\}$. For each i with $1 \leq i \leq k$, let $a_i = s(\mathsf{y}_i)$, and let $b = h_{\varphi,\mathsf{x}}(a_1, a_2, \ldots, a_k) \in A$. Since $(\mathcal{M}, a_1, a_2, \ldots, a_k) \vDash \exists \mathsf{x}\varphi$, we have $(\mathcal{M}, a_1, a_2, \ldots, a_k, b) \vDash \varphi$ by definition of $h_{\varphi,\mathsf{x}}$, so there exists $a \in A$ such that $(\mathcal{M}, s[\mathsf{x} \Rightarrow a]) \vDash \varphi$.

Therefore, we have $\mathcal{A} \preceq \mathcal{M}$. □

Corollary 4.5.5. *Suppose that $\mathcal{L}$ is countable and that $\mathcal{M}$ is an $\mathcal{L}$-structure. There exists a countable $\mathcal{L}$-structure $\mathcal{N}$ such that $\mathcal{N} \equiv \mathcal{M}$.*

Proof. Applying Theorem 4.5.4 with $X = \emptyset$, we can fix a countable elementary substructure $\mathcal{N} \preceq \mathcal{M}$. For any $\sigma \in Sent_{\mathcal{L}}$, we then have that $\mathcal{N} \vDash \sigma$ if and only if $\mathcal{M} \vDash \sigma$, so $\mathcal{N} \equiv \mathcal{M}$. □

This is our first indication that first-order logic is not powerful enough to distinguish certain aspects of cardinality, and we'll see more examples of this phenomenon after the Compactness Theorem (for first-order logic) and once we talk about infinite cardinalities and extend the Lowenheim-Skolem-Tarski result. But our first major result already has some interesting consequences.

Corollary 4.5.6. *Let $\mathcal{L}$ be a countable language. Every nonempty elementary class of $\mathcal{L}$-structures contains a structure that is countable.*

Proof. Let $\mathcal{K}$ be a nonempty elementary class of $\mathcal{L}$-structures. Since $\mathcal{K}$ is nonempty, we can fix some $\mathcal{M} \in \mathcal{K}$. By Corollary 4.5.5, we can fix an $\mathcal{N} \equiv \mathcal{M}$. Since $\mathcal{K}$ is an elementary class, it is clearly closed under elementary equivalence, so $\mathcal{N} \in \mathcal{K}$ □

We can use our corollary to argue that some classes of structures are not elementary classes. For example, you may be familiar with the result that the real numbers form the unique (up to isomorphism) Dedekind-complete ordered field. In particular, every Dedekind-complete ordered field is uncountable. It follows the class of Dedekind-complete ordered fields is not an elementary class in any countable language. In particular, it is not an elementary class in the language $\mathcal{L} = \{\mathsf{0}, \mathsf{1}, +, -, \cdot, <\}$ of ordered rings.

In Section 4.3, we introduced one way to define subobjects and maps for structures. In many cases, these definitions agree with the usual concepts of subobjects and maps (morphisms) in the corresponding area of mathematics. However, as we saw in Theorem 4.3.6, substructures and homomorphisms (or embeddings) do not typically preserve the truth of formulas, and hence may not be the "correct" definitions from the viewpoint of logic. We introduced elementary substructures to fix this issue for subobjects, and we now introduce the corresponding concept of an elementary map.

Definition 4.5.7. *Let $\mathcal{L}$ be a language and let $\mathcal{M}$ and $\mathcal{N}$ be $\mathcal{L}$-structures. An* elementary map*, or* elementary embedding*, from $\mathcal{M}$ to $\mathcal{N}$ is a function $h\colon M \to N$ such that for all $s\colon Var \to M$ and all $\varphi \in Form_{\mathcal{L}}$, we have*

$$(\mathcal{M}, s) \vDash \varphi \text{ if and only if } (\mathcal{N}, h \circ s) \vDash \varphi.$$

Written in our other notation, a function $h\colon M \to N$ is an elementary map if for all $\varphi(\mathsf{x}_1, \mathsf{x}_2, \ldots, \mathsf{x}_n) \in Form_{\mathcal{L}}$ and all $a_1, a_2, \ldots, a_n \in M$, we have

$$(\mathcal{M}, a_1, a_2, \ldots, a_n) \vDash \varphi \text{ if and only if } (\mathcal{N}, h(a_1), h(a_2), \ldots, h(a_n)) \vDash \varphi.$$

Note that we did not assume that h was injective in the above definition, but we do obtain it for free. In fact, every elementary map is an embedding, which explains the alternative name.

Proposition 4.5.8. *Let $\mathcal{L}$ be a language and let $\mathcal{M}$ and $\mathcal{N}$ be $\mathcal{L}$-structures. If $h\colon M \to N$ is an elementary map, then h is an embedding.*

Proof. Let $h\colon M \to N$ be an elementary map. We need to show that h is injective and that h is a homomorphism.

- h is injective: Let $a, b \in M$ be arbitrary with $a \neq b$. Notice that $(\mathcal{M}, a, b) \vDash \neg(\mathsf{x} = \mathsf{y})$, so $(\mathcal{N}, h(a), h(b)) \vDash \neg(\mathsf{x} = \mathsf{y})$, and hence $h(a) \neq h(b)$. Therefore, h is injective.
- h preserves constants: Let $\mathsf{c} \in \mathcal{C}$ be a constant symbol of $\mathcal{L}$. We then have that $(\mathcal{M}, \mathsf{c}^{\mathcal{M}}) \vDash \mathsf{x} = \mathsf{c}$, so $(\mathcal{N}, h(\mathsf{c}^{\mathcal{M}})) \vDash \mathsf{x} = \mathsf{c}$, and hence $h(\mathsf{c}^{\mathcal{M}}) = \mathsf{c}^{\mathcal{N}}$.
- h preserves relations: Let $\mathsf{R} \in \mathcal{R}_k$ be a relation symbol of $\mathcal{L}$, and let $a_1, a_2, \ldots, a_k \in M$ be arbitrary. Notice that

$$\begin{aligned}(a_1, a_2, \ldots, a_k) \in \mathsf{R}^{\mathcal{M}} &\Leftrightarrow (\mathcal{M}, a_1, a_2, \ldots, a_k) \vDash \mathsf{R}\mathsf{x}_1\mathsf{x}_2\ldots\mathsf{x}_k \\ &\Leftrightarrow (\mathcal{N}, h(a_1), h(a_2), \ldots, h(a_k)) \vDash \mathsf{R}\mathsf{x}_1\mathsf{x}_2\ldots\mathsf{x}_k \\ &\Leftrightarrow (h(a_1), h(a_2), \ldots, h(a_k)) \in \mathsf{R}^{\mathcal{N}}.\end{aligned}$$

- h preserves functions: Let $\mathsf{f} \in \mathcal{F}_k$ be a function symbol of $\mathcal{L}_M$, and let $a_1, a_2, \ldots, a_k \in M$ be arbitrary. We then have
$$(\mathcal{M}, a_1, a_2, \ldots, a_k, \mathsf{f}^{\mathcal{M}}(a_1, a_2, \ldots, a_k)) \vDash \mathsf{f}\mathsf{x}_1\mathsf{x}_2\ldots\mathsf{x}_k = \mathsf{x}_{k+1},$$
so
$$(\mathcal{N}, h(a_1), h(a_2), \ldots, h(a_k), h(\mathsf{f}^{\mathcal{M}}(a_1, a_2, \ldots, a_k))) \vDash \mathsf{f}\mathsf{x}_1\mathsf{x}_2\ldots\mathsf{x}_k = \mathsf{x}_{k+1},$$
and hence
$$\mathsf{f}^{\mathcal{N}}(h(a_1), h(a_2), \ldots, h(a_k)) = h(\mathsf{f}^{\mathcal{M}}(a_1, a_2, \ldots, a_k)).$$

□

The next simple result is the analogue of Proposition 4.3.5.

Proposition 4.5.9. *Let $\mathcal{L}$ be a language, let $\mathcal{M}$ be an $\mathcal{L}$-structure, and let $\mathcal{A}$ be a substructure of $\mathcal{M}$. We then have that $\mathcal{A}$ is an elementary substructure of $\mathcal{M}$ if and only if the inclusion function $i\colon A \to M$ given by $i(a) = a$ is an elementary embedding.*

Proof. Immediate from the corresponding definitions. □

4.6 Substitution

Given an $\mathcal{L}$-structure together with a variable assignment $s\colon Var \to M$, we know that every term names an element of M. Specifically, the term t names the element $\overline{s}(t)$. In normal mathematical practice, if we know that a given statement is true for all elements of a set, then we can invoke the universal quantifier on any specific element. To make this idea precise, we want to think about "substituting" a term t for a variable.

Roughly, one might naturally think that if $\forall \mathsf{x}\varphi$ is true $(\mathcal{M}, s)$, then upon taking a term t and substituting it in for x in the formula φ, the resulting formula would also be true in $(\mathcal{M}, s)$. We need a way to relate truth before substituting with truth after substituting. The hope would be the following, where we use the notation φ^t_{x} to intuitively mean that we substitute t for all free occurrences of x:

Hope 4.6.1. *Let $\mathcal{M}$ be an $\mathcal{L}$-structure, let $s\colon Var \to M$, let $t \in Term_{\mathcal{L}}$, and let $\mathsf{x} \in Var$. For all $\varphi \in Form_{\mathcal{L}}$, we have*

$$(\mathcal{M}, s) \vDash \varphi^t_{\mathsf{x}} \text{ if and only if } (\mathcal{M}, s[\mathsf{x} \Rightarrow \overline{s}(t)]) \vDash \varphi.$$

However, even with the correct definition of substitution, the above statement is not true. To see this, consider the language $\mathcal{L} = \emptyset$, and let $\varphi(\mathsf{x}) \in Form_{\mathcal{L}}$ be the formula

$$\exists \mathsf{y} \neg(\mathsf{y} = \mathsf{x}).$$

For any $\mathcal{L}$-structure $\mathcal{M}$ and any $s\colon Var \to M$, we have $(\mathcal{M}, s) \vDash \varphi$ if and only if $|M| \geq 2$. Now notice that the formula $\varphi^{\mathsf{y}}_{\mathsf{x}}$ is

$$\exists \mathsf{y} \neg(\mathsf{y} = \mathsf{y}),$$

so for any $\mathcal{L}$-structure $\mathcal{M}$ and any $s\colon Var \to M$, we have $(\mathcal{M}, s) \nvDash \varphi^{\mathsf{y}}_{\mathsf{x}}$. Therefore, the above hope fails whenever $\mathcal{M}$ is an $\mathcal{L}$-structure with $|M| \geq 2$. The problem is that the term we substituted (in this case y) had a variable which became "captured" by a quantifier, resulting in a fundamental change of the "meaning" of the formula.

Before diving into how to avoid the underlying issue present in the above example, we first formally define substitution for terms and show that it behaves as expected.

Definition 4.6.2. *Let $\mathcal{L}$ be a language, let $\mathsf{x} \in Var$, and let $t \in Term_{\mathcal{L}}$. Define a function $Subst^t_{\mathsf{x}}\colon Term_{\mathcal{L}} \to Term_{\mathcal{L}}$, where we use u^t_{x} to denote $Subst^t_{\mathsf{x}}(u)$, as follows:*

1. $\mathsf{c}^t_{\mathsf{x}} = \mathsf{c}$ *for all* $\mathsf{c} \in \mathcal{C}$.
2. $\mathsf{y}^t_{\mathsf{x}} = \begin{cases} t & \text{if } \mathsf{y} = \mathsf{x} \\ \mathsf{y} & \text{otherwise} \end{cases}$
   for all $\mathsf{y} \in Var$.
3. $(\mathsf{f}u_1u_2\dots u_k)^t_{\mathsf{x}} = \mathsf{f}(u_1)^t_{\mathsf{x}}(u_2)^t_{\mathsf{x}}\cdots(u_k)^t_{\mathsf{x}}$ *for all* $\mathsf{f} \in \mathcal{F}_k$ *and all* $u_1, u_2, \dots, u_k \in Term_{\mathcal{L}}$.

When substituting a term for a variable, here is the key lemma that relates how to interpret a term before and after the substitution.

Lemma 4.6.3. *Let $\mathcal{M}$ be an $\mathcal{L}$-structure, let $s\colon Var \to M$, let $t \in Term_{\mathcal{L}}$, and let $\mathsf{x} \in Var$. For all $u \in Term_{\mathcal{L}}$, we have*

$$\overline{s}(u^t_{\mathsf{x}}) = \overline{s[\mathsf{x} \Rightarrow \overline{s}(t)]}(u).$$

Although the statement of the lemma is symbol heavy, it expresses something quite natural. In order to determine the value of the term u^t_{x} according to the variable assignment imposed by s, we need only change s so that x now gets sent to $\overline{s}(t)$ (the value of t assigned by s), and evaluate u using this new variable assignment.

Proof. The proof is by induction on $Term_{\mathcal{L}}$. For any $\mathsf{c} \in \mathcal{C}$, we have

$$\begin{aligned} \overline{s}(\mathsf{c}^t_{\mathsf{x}}) &= \overline{s}(\mathsf{c}) \\ &= \mathsf{c}^{\mathcal{M}} \\ &= s[\mathsf{x} \Rightarrow \overline{s}(t)](\mathsf{c}) \\ &= \overline{s[\mathsf{x} \Rightarrow \overline{s}(t)]}(\mathsf{c}). \end{aligned}$$

We now handle the case where $u \in Var$. If $u = \mathsf{x}$, then

$$\begin{aligned}\overline{s}(\mathsf{x}^t_{\mathsf{x}}) &= \overline{s}(t)\\ &= s[\mathsf{x} \Rightarrow \overline{s}(t)](\mathsf{x})\\ &= \overline{s[\mathsf{x} \Rightarrow \overline{s}(t)]}(\mathsf{x}).\end{aligned}$$

On the other hand, if $u = \mathsf{y} \in Var$ and $\mathsf{y} \neq \mathsf{x}$, then we have

$$\begin{aligned}\overline{s}(\mathsf{y}^t_{\mathsf{x}}) &= \overline{s}(\mathsf{y})\\ &= s(\mathsf{y})\\ &= \mathsf{y}\\ &= s[\mathsf{x} \Rightarrow \overline{s}(t)](\mathsf{y})\\ &= \overline{s[\mathsf{x} \Rightarrow \overline{s}(t)]}(\mathsf{y}).\end{aligned}$$

Finally, let $\mathsf{f} \in \mathcal{F}_k$ be arbitrary, and that the statement is true for $u_1, u_2, \ldots, u_k \in Term_{\mathcal{L}}$. We then have

$$\begin{aligned}\overline{s}((\mathsf{f}u_1u_2\cdots u_k)^t_{\mathsf{x}}) &= \overline{s}(\mathsf{f}(u_1)^t_{\mathsf{x}}(u_2)^t_{\mathsf{x}}\cdots(u_k)^t_{\mathsf{x}})\\ &= \mathsf{f}^{\mathcal{M}}(\overline{s}((u_1)^t_{\mathsf{x}}), \overline{s}((u_2)^t_{\mathsf{x}}), \ldots, \overline{s}((u_k)^t_{\mathsf{x}}))\\ &= \mathsf{f}^{\mathcal{M}}(\overline{s[\mathsf{x} \Rightarrow \overline{s}(t)]}(u_1), \overline{s[\mathsf{x} \Rightarrow \overline{s}(t)]}(u_2), \ldots, \overline{s[\mathsf{x} \Rightarrow \overline{s}(t)]}(u_k)) \qquad \text{(by induction)}\\ &= \overline{s[\mathsf{x} \Rightarrow \overline{s}(t)]}(\mathsf{f}u_1u_2\cdots u_k)\end{aligned}$$

Therefore, the statement is true for $\mathsf{f}u_1u_2\cdots u_k$.

This completes the induction. □

Now that we have handled terms, we move to define substitution for formulas. In the case of terms, we naturally replaced every occurrence of x with the term t. However, we do have to be a bit more discriminating when faced with formulas. For example, we certainly do not want to change $\forall \mathsf{x}\varphi$ into $\forall t\varphi$ (which would not even be a formula if $t \notin Var$), nor do we want to mess with an x inside the scope of such a quantifier. We thus make the following recursive definition. Intuitively, we want to replace each *free* occurrence of x with the variable t. We now turn that intuition into a precise recursive definition.

Definition 4.6.4. *Let $\mathcal{L}$ be a language, let $t \in Term_{\mathcal{L}}$, and let $\mathsf{x} \in Var$. Define $FreeSubst^t_{\mathsf{x}}\colon Form_{\mathcal{L}} \to Form_{\mathcal{L}}$, again denoted φ^t_{x}, as follows:*

1. $(\mathsf{R}u_1u_2\cdots u_k)^t_{\mathsf{x}} = \mathsf{R}(u_1)^t_{\mathsf{x}}(u_2)^t_{\mathsf{x}}\cdots(u_k)^{t_{\mathsf{x}}}$ *for all $\mathsf{R} \in \mathcal{R}_k$ and all $u_1, u_2, \ldots, u_k \in Term_{\mathcal{L}}$.*
2. *We define $(= u_1u_2)^t_{\mathsf{x}}$ to be $= (u_1)^t_{\mathsf{x}}(u_2)^t_{\mathsf{x}}$ for all $u_1, u_2 \in Term_{\mathcal{L}}$.*
3. $(\neg\varphi)^t_{\mathsf{x}} = \neg(\varphi^t_{\mathsf{x}})$ *for all $\varphi \in Form_{\mathcal{L}}$.*
4. $(\Diamond\varphi\psi)^t_{\mathsf{x}} = \Diamond\varphi^t_{\mathsf{x}}\psi^t_{\mathsf{x}}$ *for all $\varphi, \psi \in Form_{\mathcal{L}}$ and all $\Diamond \in \{\wedge, \vee, \to\}$.*
5. $(\mathsf{Q}\mathsf{y}\varphi)^t_{\mathsf{x}} = \begin{cases}\mathsf{Q}\mathsf{y}\varphi & \text{if } \mathsf{x} = \mathsf{y}\\ \mathsf{Q}\mathsf{y}(\varphi^t_{\mathsf{x}}) & \text{otherwise}\end{cases}$

   for all $\varphi \in Form_{\mathcal{L}}$, $\mathsf{y} \in Var$, and $\mathsf{Q} \in \{\exists, \forall\}$.

Now that we have formally defined (free) substitution, we can turn our attention to the issue raised by the example at the beginning of this section. When we substitute a term t in for the free occurrences of a variable x, it is possible that a variable appearing in t will be "captured" by a quantifier. We will now formally define the substitutions that do *not* result in such a captured variable.

Definition 4.6.5. *Let $\mathcal{L}$ be a language, let $t \in Term_{\mathcal{L}}$, and let $\mathsf{x} \in Var$. We define $ValidSubst^t_{\mathsf{x}}\colon Form_{\mathcal{L}} \to \{0,1\}$ as follows:*

1. $ValidSubst^t_{\mathsf{x}}(\varphi) = 1$ *for all* $\varphi \in AtomicForm_{\mathcal{L}}$.

2. $ValidSubst^t_{\mathsf{x}}(\neg\varphi) = ValidSubst^t_{\mathsf{x}}(\varphi)$ *for all* $\varphi \in Form_{\mathcal{L}}$.

3. $$ValidSubst^t_{\mathsf{x}}(\diamond\varphi\psi) = \begin{cases} 1 & \text{if } ValidSubst^t_{\mathsf{x}}(\varphi) = 1 \text{ and } ValidSubst^t_{\mathsf{x}}(\psi) = 1 \\ 0 & \text{otherwise} \end{cases}$$
   for all $\varphi, \psi \in Form_{\mathcal{L}}$ *and all* $\diamond \in \{\wedge, \vee, \to\}$.

4. $$ValidSubst^t_{\mathsf{x}}(\mathsf{Q}\mathsf{y}\varphi) = \begin{cases} 1 & \text{if } \mathsf{x} \notin FreeVar(\mathsf{Q}\mathsf{y}\varphi) \\ 1 & \text{if } \mathsf{y} \notin OccurVar(t) \text{ and } ValidSubst^t_{\mathsf{x}}(\varphi) = 1 \\ 0 & \text{otherwise} \end{cases}$$
   for all $\varphi \in Form_{\mathcal{L}}$, $\mathsf{y} \in Var$, *and* $\mathsf{Q} \in \{\forall, \exists\}$.

Before proving the fundamental Substitution Theorem below, we first establish a simple result to demonstrate how we can work with the recursive definitions.

Proposition 4.6.6. *Let $\mathcal{L}$ be a language, let $t \in Term_{\mathcal{L}}$, and let $\mathsf{x} \in Var$. For all $\varphi \in Form_{\mathcal{L}}$ with $OccurVar(t) \cap BoundVar(\varphi) = \emptyset$, we have $ValidSubst^t_{\mathsf{x}}(\varphi) = 1$.*

Proof. The proof is by induction on φ.

- For the base case when $\varphi \in AtomicForm_{\mathcal{L}}$, we trivially have $ValidSubst^t_{\mathsf{x}}(\varphi) = 1$ by definition.

- Suppose that the statement is true for φ and that $OccurVar(t) \cap BoundVar(\neg\varphi) = \emptyset$. By definition, we have $BoundVar(\neg\varphi) = BoundVar(\varphi)$, and hence $OccurVar(t) \cap BoundVar(\varphi) = \emptyset$. By induction, we know that $ValidSubst^t_{\mathsf{x}}(\varphi) = 1$, so $ValidSubst^t_{\mathsf{x}}(\neg\varphi) = 1$ by definition.

- Suppose that the statement is true for both φ and ψ, that $\diamond \in \{\wedge, \vee, \to\}$, and that $OccurVar(t) \cap BoundVar(\diamond\varphi\psi) = \emptyset$. By definition, we have $BoundVar(\diamond\varphi\psi) = BoundVar(\varphi) \cup BoundVar(\psi)$, and hence both $OccurVar(t) \cap BoundVar(\varphi) = \emptyset$ and $OccurVar(t) \cap BoundVar(\psi) = \emptyset$. By induction, we know that both $ValidSubst^t_{\mathsf{x}}(\varphi) = 1$ and $ValidSubst^t_{\mathsf{x}}(\psi) = 1$, so $ValidSubst^t_{\mathsf{x}}(\diamond\varphi\psi) = 1$ by definition.

- Suppose that the statement is true for φ and that $OccurVar(t) \cap BoundVar(\mathsf{Q}\mathsf{y}\varphi) = \emptyset$, where $\mathsf{y} \in Var$ and $\mathsf{Q} \in \{\forall, \exists\}$. By definition, we have $BoundVar(\mathsf{Q}\mathsf{y}\varphi) = BoundVar(\varphi) \cup \{\mathsf{y}\}$, and hence we have both $\mathsf{y} \notin OccurVar(t)$ and also $OccurVar(t) \cap BoundVar(\varphi) = \emptyset$. By induction, we know that $ValidSubst^t_{\mathsf{x}}(\varphi) = 1$, so $ValidSubst^t_{\mathsf{x}}(\mathsf{Q}\mathsf{y}\varphi) = 1$ by definition.

□

We now prove the correct analogue of the hope expressed at the beginning of the section. That is, we show that under the assumption that $ValidSubst^t_{\mathsf{x}}(\varphi) = 1$, we can directly relation the the truth of φ^t_{x} in a structure to the truth of φ by changing the variable assignment on x appropriately.

Theorem 4.6.7 (Substitution Theorem)**.** *Let $\mathcal{M}$ be an $\mathcal{L}$-structure, let $s\colon Var \to M$, let $t \in Term_{\mathcal{L}}$, and let $\mathsf{x} \in Var$. For all $\varphi \in Form_{\mathcal{L}}$ with $ValidSubst^t_{\mathsf{x}}(\varphi) = 1$, we have*

$$(\mathcal{M}, s) \vDash \varphi^t_{\mathsf{x}} \text{ if and only if } (\mathcal{M}, s[\mathsf{x} \Rightarrow \overline{s}(t)]) \vDash \varphi.$$

Proof. The proof is by induction on φ. We first handle the case when $\varphi \in AtomicForm_{\mathcal{L}}$. Suppose that $\mathsf{R} \in \mathcal{R}_k$ and that $u_1, u_2, \ldots, u_k \in Term_{\mathcal{L}}$. We then have

$$\begin{aligned}
(\mathcal{M}, s) \vDash (\mathsf{R}u_1u_2\cdots u_k)^t_{\mathsf{x}} &\Leftrightarrow (\mathcal{M}, s) \vDash \mathsf{R}(u_1)^t_{\mathsf{x}}(u_2)^t_{\mathsf{x}}\cdots(u_k)^t_{\mathsf{x}} \\
&\Leftrightarrow (\overline{s}((u_1)^t_{\mathsf{x}}), \overline{s}((u_2)^t_{\mathsf{x}}), \cdots, \overline{s}((u_k)^t_{\mathsf{x}})) \in \mathsf{R}^{\mathcal{M}} \\
&\Leftrightarrow (\overline{s[\mathsf{x} \Rightarrow \overline{s}(t)]}(u_1), \overline{s[\mathsf{x} \Rightarrow \overline{s}(t)]}(u_2), \cdots, \overline{s[\mathsf{x} \Rightarrow \overline{s}(t)]}(u_k)) \in \mathsf{R}^{\mathcal{M}} \quad \text{(by Lemma 4.6.3)} \\
&\Leftrightarrow (\mathcal{M}, s[\mathsf{x} \Rightarrow \overline{s}(t)]) \vDash \mathsf{R}u_1u_2\cdots u_k.
\end{aligned}$$

If $u_1, u_2 \in Term_{\mathcal{L}}$, we have

$$\begin{aligned}
(\mathcal{M}, s) \vDash (= u_1u_2)^t_{\mathsf{x}} &\Leftrightarrow (\mathcal{M}, s) \vDash = (u_1)^t_{\mathsf{x}}(u_2)^t_{\mathsf{x}} \\
&\Leftrightarrow \overline{s}((u_1)^t_{\mathsf{x}}) = \overline{s}((u_2)^t_{\mathsf{x}}) \\
&\Leftrightarrow \overline{s[\mathsf{x} \Rightarrow \overline{s}(t)]}(u_1) = \overline{s[\mathsf{x} \Rightarrow \overline{s}(t)]}(u_2) \qquad \text{(by Lemma 4.6.3)} \\
&\Leftrightarrow (\mathcal{M}, s[\mathsf{x} \Rightarrow \overline{s}(t)]) \vDash = u_1u_2.
\end{aligned}$$

Suppose that the statement is true for φ and that $ValidSubst^t_{\mathsf{x}}(\neg\varphi) = 1$. We then have that $ValidSubst^t_{\mathsf{x}}(\varphi) = 1$, and hence

$$\begin{aligned}
(\mathcal{M}, s) \vDash (\neg\varphi)^t_{\mathsf{x}} &\Leftrightarrow (\mathcal{M}, s) \vDash \neg(\varphi^t_{\mathsf{x}}) \\
&\Leftrightarrow (\mathcal{M}, s) \nvDash \varphi^t_{\mathsf{x}} \\
&\Leftrightarrow (\mathcal{M}, s[\mathsf{x} \Rightarrow \overline{s}(t)]) \nvDash \varphi \qquad \text{(by induction)} \\
&\Leftrightarrow (\mathcal{M}, s[\mathsf{x} \Rightarrow \overline{s}(t)]) \vDash \neg\varphi.
\end{aligned}$$

The connectives $\wedge$,$\vee$, and $\rightarrow$ are similarly uninteresting.

We next handle the existential quantifier. Suppose that the statement is true for φ, that $\mathsf{y} \in Var$, and that $ValidSubst^t_{\mathsf{x}}(\exists\mathsf{y}\varphi) = 1$. By definition of $ValidSubst^t_{\mathsf{x}}$, we have two cases. If $\mathsf{x} \notin FreeVar(\exists\mathsf{y}\varphi)$, then we have

$$\begin{aligned}
(\mathcal{M}, s) \vDash (\exists\mathsf{y}\varphi)^t_{\mathsf{x}} &\Leftrightarrow (\mathcal{M}, s) \vDash \exists\mathsf{y}\varphi \\
&\Leftrightarrow (\mathcal{M}, s[\mathsf{x} \Rightarrow \overline{s}(t)]) \vDash \exists\mathsf{y}\varphi \qquad \text{(by Proposition 4.2.6).}
\end{aligned}$$

Otherwise, we have $\mathsf{y} \notin OccurVar(t)$ and $ValidSubst^t_{\mathsf{x}}(\varphi) = 1$. Notice that since $\mathsf{y} \notin OccurVar(t)$, we have $\overline{s[\mathsf{y} \Rightarrow a]}(t) = \overline{s}(t)$ by Proposition 4.2.6. Also, since we are not in the first case, we have $\mathsf{x} \in FreeVar(\exists\mathsf{y}\varphi)$, and in particular $\mathsf{x} \neq \mathsf{y}$. Therefore,

$$\begin{aligned}
(\mathcal{M}, s) \vDash (\exists\mathsf{y}\varphi)^t_{\mathsf{x}} &\Leftrightarrow (\mathcal{M}, s) \vDash \exists\mathsf{y}(\varphi^t_{\mathsf{x}}) && \text{(since } \mathsf{x} \neq \mathsf{y}\text{)} \\
&\Leftrightarrow \text{There exists } a \in M \text{ such that } (\mathcal{M}, s[\mathsf{y} \Rightarrow a]) \vDash \varphi^t_{\mathsf{x}} \\
&\Leftrightarrow \text{There exists } a \in M \text{ such that } (\mathcal{M}, (s[\mathsf{y} \Rightarrow a])[\mathsf{x} \Rightarrow \overline{s[\mathsf{y} \Rightarrow a]}(t)]) \vDash \varphi && \text{(by induction)} \\
&\Leftrightarrow \text{There exists } a \in M \text{ such that } (\mathcal{M}, (s[\mathsf{y} \Rightarrow a])[\mathsf{x} \Rightarrow \overline{s}(t)]) \vDash \varphi && \text{(from above)} \\
&\Leftrightarrow \text{There exists } a \in M \text{ such that } (\mathcal{M}, (s[\mathsf{x} \Rightarrow \overline{s}(t)])[\mathsf{y} \Rightarrow a]) \vDash \varphi && \text{(since } \mathsf{x} \neq \mathsf{y}\text{)} \\
&\Leftrightarrow (\mathcal{M}, s[\mathsf{x} \Rightarrow \overline{s}(t)]) \vDash \exists\mathsf{y}\varphi.
\end{aligned}$$

We finally handle the universal quantifier. Suppose that the statement is true for φ, that $\mathsf{y} \in Var$, and that $ValidSubst^t_{\mathsf{x}}(\forall\mathsf{y}\varphi) = 1$. By definition of $ValidSubst^t_{\mathsf{x}}$, we have two cases. If $\mathsf{x} \notin FreeVar(\forall\mathsf{y}\varphi)$, then we have

$$\begin{aligned}
(\mathcal{M}, s) \vDash (\forall\mathsf{y}\varphi)^t_{\mathsf{x}} &\Leftrightarrow (\mathcal{M}, s) \vDash \forall\mathsf{y}\varphi \\
&\Leftrightarrow (\mathcal{M}, s[\mathsf{x} \Rightarrow \overline{s}(t)]) \vDash \forall\mathsf{y}\varphi \qquad \text{(by Proposition 4.2.6).}
\end{aligned}$$

Otherwise, we have $\mathsf{y} \notin OccurVar(t)$ and $ValidSubst^t_{\mathsf{x}}(\varphi) = 1$. Notice that since $\mathsf{y} \notin OccurVar(t)$, we have $\overline{s[\mathsf{y} \Rightarrow a]}(t) = \overline{s}(t)$ by Proposition 4.2.6. Also, since we are not in the first case, we have $\mathsf{x} \in FreeVar(\exists \mathsf{y}\varphi)$, and in particular $\mathsf{x} \neq \mathsf{y}$. Therefore,

$$\begin{aligned}
(\mathcal{M}, s) \vDash (\forall \mathsf{y}\varphi)^t_{\mathsf{x}} &\Leftrightarrow (\mathcal{M}, s) \vDash \forall \mathsf{y}(\varphi^t_{\mathsf{x}}) && \text{(since } \mathsf{x} \neq \mathsf{y}\text{)} \\
&\Leftrightarrow \text{For all } a \in M, \text{ we have } (\mathcal{M}, s[\mathsf{y} \Rightarrow a]) \vDash \varphi^t_{\mathsf{x}} \\
&\Leftrightarrow \text{For all } a \in M, \text{ we have } (\mathcal{M}, (s[\mathsf{y} \Rightarrow a])[\mathsf{x} \Rightarrow \overline{s[\mathsf{y} \Rightarrow a]}(t)]) \vDash \varphi && \text{(by induction)} \\
&\Leftrightarrow \text{For all } a \in M, \text{ we have } (\mathcal{M}, (s[\mathsf{y} \Rightarrow a])[\mathsf{x} \Rightarrow \overline{s}(t)]) \vDash \varphi && \text{(from above)} \\
&\Leftrightarrow \text{For all } a \in M, \text{ we have } (\mathcal{M}, (s[\mathsf{x} \Rightarrow \overline{s}(t)])[\mathsf{y} \Rightarrow a]) \vDash \varphi && \text{(since } \mathsf{x} \neq \mathsf{y}\text{)} \\
&\Leftrightarrow (\mathcal{M}, s[\mathsf{x} \Rightarrow \overline{s}(t)]) \vDash \forall \mathsf{y}\varphi.
\end{aligned}$$

□

As mentioned at the beginning of the section, it is natural to think that if $\forall \mathsf{x}\varphi$ is true $(\mathcal{M}, s)$, then upon taking a term t and substituting it in for x in the formula φ, the resulting formula would also be true in $(\mathcal{M}, s)$. Using the Substitution Theorem we can prove this result (and a corresponding result about existential quantifiers) under the assumption that $ValidSubst^t_{\mathsf{x}}(\varphi) = 1$.

Corollary 4.6.8. *Let $\mathcal{M}$ be an $\mathcal{L}$-structure, let $s\colon Var \to M$ be a variable assignment, let $\mathsf{x} \in Var$, and let $t \in Term_{\mathcal{L}}$.*

1. *If $(\mathcal{M}, s) \vDash \forall \mathsf{x}\varphi$ and $ValidSubst^t_{\mathsf{x}}(\varphi) = 1$, then $(\mathcal{M}, s) \vDash \varphi^t_{\mathsf{x}}$.*
2. *If $(\mathcal{M}, s) \vDash \varphi^t_{\mathsf{x}}$ and $ValidSubst^t_{\mathsf{x}}(\varphi) = 1$, then $(\mathcal{M}, s) \vDash \exists \mathsf{x}\varphi$.*

Proof.

1. Assume that $(\mathcal{M}, s) \vDash \forall \mathsf{x}\varphi$ and $ValidSubst^t_{\mathsf{x}}(\varphi) = 1$. Since $(\mathcal{M}, s) \vDash \forall \mathsf{x}\varphi$, we know by the recursive definition of $\vDash$ that $(\mathcal{M}, s[\mathsf{x} \Rightarrow a]) \vDash \varphi$ for all $a \in M$. In particular, since $\overline{s}(t) \in M$, it follows that $(\mathcal{M}, s[\mathsf{x} \Rightarrow \overline{s}(t)]) \vDash \varphi$. Using the Substitution Theorem, we conclude that $(\mathcal{M}, s) \vDash \varphi^t_{\mathsf{x}}$.
2. Assume that $(\mathcal{M}, s) \vDash \varphi^t_{\mathsf{x}}$ and $ValidSubst^t_{\mathsf{x}}(\varphi) = 1$. Using the Substitution Theorem, we conclude that $(\mathcal{M}, s[\mathsf{x} \Rightarrow \overline{s}(t)]) \vDash \varphi$. Since $\overline{s}(t) \in M$, it follows that there exists $a \in M$ with $(\mathcal{M}, s[\mathsf{x} \Rightarrow a]) \vDash \varphi$. Therefore, $(\mathcal{M}, s) \vDash \exists \mathsf{x}\varphi$.

□

What if we want to substitute terms $t_1, t_2, \ldots, t_n$ for (distinct) variables $\mathsf{x}_1, \mathsf{x}_2, \ldots, \mathsf{x}_n$? We can of course accomplish this in stages by first substituting t_1 for x_1, then substituting t_2 for x_2, etc. Although this is a perfectly reasonable approach, one issue is that the order in which we perform this iterated substitution can matter! To see this, notice that if $\mathsf{x}_2 \in OccurVar(t_1)$, then after substituting t_1 for x_1, we have introduced new occurrences of x_2, which will then be replaced by t_2 under the next substitution. For example, consider the language $\mathcal{L} = \{\mathsf{c}, \mathsf{R}\}$ where c is a constant symbol and R is a binary relation symbol, and let φ be the formula $\mathsf{R}\mathsf{x}_1\mathsf{x}_2$. Let $t_1 = \mathsf{x}_2$ and let $t_2 = \mathsf{c}$ We then have

$$((\mathsf{R}\mathsf{x}_1\mathsf{x}_2)^{\mathsf{x}_2}_{\mathsf{x}_1})^{\mathsf{c}}_{\mathsf{x}_2} = (\mathsf{R}\mathsf{x}_2\mathsf{x}_2)^{\mathsf{c}}_{\mathsf{x}_2} = \mathsf{R}\mathsf{c}\mathsf{c}$$

while

$$((\mathsf{R}\mathsf{x}_1\mathsf{x}_2)^{\mathsf{c}}_{\mathsf{x}_2})^{\mathsf{x}_2}_{\mathsf{x}_1} = (\mathsf{R}\mathsf{x}_1\mathsf{c})^{\mathsf{x}_2}_{\mathsf{x}_1} = \mathsf{R}\mathsf{x}_2\mathsf{c}.$$

In order to avoid this problem, we instead think about performing *simultaneous* substitution rather than *iterative* substitution. That is, we think about instantly replacing each (free occurrence of) x_i by the corresponding t_i. For our purposes, simultaneous substitution will be more natural and elegant, although we will

typically use it in the case where the t_i do not contain variables (so simultaneous and iterative approaches would lead to the same result).

We now formally define this idea by staring with a function $\alpha\colon Var \to Term_{\mathcal{L}}$, which we think of as coding the simultaneous substitution of replacing each variable x by the term $\alpha(\mathsf{x})$. This approach permits the simultaneous substitution of infinitely many variables (although any given formula will only contain finitely many), and one thinks of setting $\alpha(\mathsf{x}) = \mathsf{x}$ in the case where we do not want to substitute anything for x. We first define substitution for terms.

Definition 4.6.9. *Let $\mathcal{L}$ be a language and let $\alpha\colon Var \to Term_{\mathcal{L}}$ be a function. We define a function $Subst^{\alpha}\colon Term_{\mathcal{L}} \to Term_{\mathcal{L}}$, where we use u^{α} to denote $Subst^{\alpha}(u)$, as follows:*

1. $\mathsf{c}^{\alpha} = \mathsf{c}$ *for all* $\mathsf{c} \in \mathcal{C}$.
2. $\mathsf{x}^{\alpha} = \alpha(\mathsf{x})$ *for all* $\mathsf{x} \in Var$.
3. $(\mathsf{f}t_1t_2\dots u_k)^{\alpha} = \mathsf{f}(u_1)^{\alpha}(u_2)^{\alpha}\cdots(u_k)^{\alpha}$ *for all* $\mathsf{f} \in \mathcal{F}_k$ *and all* $u_1, u_2, \dots, u_k \in Term_{\mathcal{L}}$.

We now state the analogue of Lemma 4.6.3. To think through the appropriate statement, consider having a function $\alpha\colon Var \to Term_{\mathcal{L}}$ together with a variable assignment $s\colon Var \to M$. Notice that $\overline{s}\colon Term_{\mathcal{L}} \to M$, so $\overline{s} \circ \alpha\colon Var \to M$ is a variable assignment that takes an input x, replaces it by the corresponding term, and then evaluates the result. Since $\overline{s} \circ \alpha$ is a variable assignment, we can form $\overline{\overline{s} \circ \alpha}\colon Term_{\mathcal{L}} \to M$. Putting together these ideas, we arrive at the following result.

Lemma 4.6.10. *Let $\mathcal{M}$ be an $\mathcal{L}$-structure, let $s\colon Var \to M$, and let $\alpha\colon Var \to Term_{\mathcal{L}}$. For all $u \in Term_{\mathcal{L}}$, we have*

$$\overline{s}(u^{\alpha}) = \overline{(\overline{s} \circ \alpha)}(u).$$

Proof. Exercise. □

We now extend our definition of simultaneous substitution to formulas. The key part of the definition is how to handle quantifiers. When faced with $(\mathsf{Q}\mathsf{x}\varphi)^{\alpha}$, we want to leave the occurrences of x inside of φ alone, and we accomplish this by changing α to a new function that sends x to itself.

Definition 4.6.11. *Let $\mathcal{L}$ be a language and let $\alpha\colon Var \to Term_{\mathcal{L}}$. We define $FreeSubst^{\alpha}\colon Form_{\mathcal{L}} \to Form_{\mathcal{L}}$, again denoted φ^{α}, as follows:*

- $(\mathsf{R}u_1u_2\cdots u_k)^{\alpha} = \mathsf{R}(u_1)^{\alpha}(u_2)^{\alpha}\cdots(u_k)^{\alpha}$ *for all* $\mathsf{R} \in \mathcal{R}_k$ *and all* $u_1, u_2, \dots, u_k \in Term_{\mathcal{L}}$.
- *We define* $(= u_1u_2)^{\alpha}$ *to be* $= (u_1)^{\alpha}(u_2)^{\alpha}$ *for all* $u_1, u_2 \in Term_{\mathcal{L}}$.
- $(\neg\varphi)^{\alpha} = \neg(\varphi^{\alpha})$ *for all* $\varphi \in Form_{\mathcal{L}}$.
- $(\diamond\varphi\psi)^{\alpha} = \diamond\varphi^{\alpha}\psi^{\alpha}$ *for all* $\varphi, \psi \in Form_{\mathcal{L}}$ *and all* $\diamond \in \{\wedge, \vee, \to\}$.
- $(\mathsf{Q}\mathsf{x}\varphi)^{\alpha} = \mathsf{Q}\mathsf{x}(\varphi^{\alpha_{\mathsf{x}}})$ *for all* $\varphi \in Form_{\mathcal{L}}$, $\mathsf{y} \in Var$, *and* $\mathsf{Q} \in \{\exists, \forall\}$, *where* $\alpha_{\mathsf{x}}\colon Var \to Term_{\mathcal{L}}$ *is given by*

$$\alpha_{\mathsf{x}}(\mathsf{z}) = \begin{cases} \alpha(\mathsf{z}) & \text{if } \mathsf{z} \neq \mathsf{x} \\ \mathsf{x} & \text{if } \mathsf{z} = \mathsf{x}. \end{cases}$$

Finally, we have to extend our definition of $ValidSubst$ to our more general concept of substitution.

Definition 4.6.12. *Let $\mathcal{L}$ be a language and let $\alpha\colon Var \to Term_{\mathcal{L}}$. We define $ValidSubst^{\alpha}\colon Form_{\mathcal{L}} \to \{0, 1\}$ as follows:*

- $ValidSubst^{\alpha}(\varphi) = 1$ *for all* $\varphi \in AtomicForm_{\mathcal{L}}$.

- $ValidSubst^{\alpha}(\neg\varphi) = ValidSubst^{\alpha}(\varphi)$ *for all* $\varphi \in Form_{\mathcal{L}}$.
- $ValidSubst^{\alpha}(\Diamond\varphi\psi) = \begin{cases} 1 & \text{if } ValidSubst^{\alpha}(\varphi) = 1 \text{ and } ValidSubst^{\alpha}(\psi) = 1 \\ 0 & \text{otherwise} \end{cases}$

  for all $\varphi, \psi \in Form_{\mathcal{L}}$ *and all* $\Diamond \in \{\wedge, \vee, \rightarrow\}$.
- $ValidSubst^{\alpha}(\mathsf{Q}\mathsf{x}\varphi) = \begin{cases} 1 & \text{if } ValidSubst^{\alpha_{\mathsf{x}}}(\varphi) = 1 \text{ and} \\ & \qquad \mathsf{x} \notin OccurVar(\alpha(\mathsf{y})) \text{ for all } \mathsf{y} \in FreeVar(\varphi)\backslash\{\mathsf{x}\} \\ 0 & \text{otherwise} \end{cases}$

  for all $\varphi \in Form_{\mathcal{L}}$, $\mathsf{x} \in Var$, *and* $\mathsf{Q} \in \{\forall, \exists\}$, *where* α_{x} *is as described in the previous definition.*

With all of the definitions in hand, we state the analogue of Theorem 4.6.7. The inductive proof is similar, but we leave the details to the reader.

Theorem 4.6.13 (Multiple Substitution Theorem)**.** *Let* $\mathcal{M}$ *be an* $\mathcal{L}$*-structure, let* $s\colon Var \rightarrow M$, *and let* $\alpha\colon Var \rightarrow Term_{\mathcal{L}}$. *For all* $\varphi \in Form_{\mathcal{L}}$ *with* $ValidSubst^{\alpha}(\varphi) = 1$, *we have*

$$(\mathcal{M}, s) \vDash \varphi^{\alpha} \text{ if and only if } (\mathcal{M}, \overline{s} \circ \alpha) \vDash \varphi.$$

Proof. Exercise. □

We will almost always apply the Multiple Substitution Theorem in the special case where $\alpha\colon Var \rightarrow Term_{\mathcal{L}}$ has the property that for all $\mathsf{z} \in Var$, either $\alpha(\mathsf{z}) = \mathsf{z}$ or $\alpha(\mathsf{z})$ contains no variables. In other words, whenever α actually encodes a actual change to a variable, it turns it into a term that is built up from constants and function symbols. We give these types of terms a special name.

Definition 4.6.14. *Let* $\mathcal{L}$ *be a language and let* $t \in Term_{\mathcal{L}}$

1. *We say that* t *is a* closed term *if* $OccurVar(t) = \emptyset$.
2. *If* t *is a closed term and* $\mathcal{M}$ *is an* $\mathcal{L}$*-structure, we use the notation* $t^{\mathcal{M}}$ *for the value of* $s(t)$ *for some (any) variable assignment* $s\colon Var \rightarrow M$ *(which is well-defined by Proposition 4.2.6).*

In other words, since a closed term does not contain any variables, we can unambiguously assign it a value in a given $\mathcal{L}$-structure without having a particular variable assignment around. In the situation where we substitute some of the free variables of a given formula with closed terms, then we arrive at the following intuitive result that will play an important role in Chapter 7.

Corollary 4.6.15. *Let* $\mathcal{M}$ *be an* $\mathcal{L}$*-structure, let* $\varphi(\mathsf{x}_1, \ldots, \mathsf{x}_k, \mathsf{y}_1, \ldots, \mathsf{y}_n) \in Form_{\mathcal{L}}$, *and let* $t_1, \ldots, t_n$ *be closed terms of* $\mathcal{L}$. *Define* $\alpha\colon Var \rightarrow Term_{\mathcal{L}}$ *by letting*

$$\alpha(\mathsf{z}) = \begin{cases} t_i & \text{if } \mathsf{z} = \mathsf{y}_i \\ \mathsf{z} & \text{otherwise.} \end{cases}$$

We then have the following:

1. $FreeVar(\varphi^{\alpha}) \subseteq \{\mathsf{x}_1, \ldots, \mathsf{x}_k\}$.
2. $ValidSubst^{\alpha}(\varphi) = 1$.
3. *For all* $a_1, \ldots, a_k \in M$, *we have* $(\mathcal{M}, a_1, \ldots, a_k, t_1^{\mathcal{M}}, \ldots, t_n^{\mathcal{M}}) \vDash \varphi$ *if and only if* $(\mathcal{M}, a_1, \ldots, a_k) \vDash \varphi^{\alpha}$.

Proof. See Exercises 10, 11 and 12 for the first two claims. The final claim then follows from Theorem 4.6.13. □

4.7 Exercises

1. (a) Let $\mathcal{L} = \{\mathsf{f}\}$ where f is a unary function symbol. Show that the class of all $\mathcal{L}$-structures $\mathcal{M}$ such that $\mathsf{f}^{\mathcal{M}}$ is a bijection on M is a strong elementary class in the language $\mathcal{L}$.

   (b) A *directed graph* is a nonempty set V of vertices together with a set $E \subseteq V \times V$ where $(u, w) \in E$ intuitively represents an edge originating at u and terminating at w. A cycle in a directed graph is a sequence $u_1 u_2 \cdots u_k$ of vertices, such that $(u_i, u_{i+1}) \in E$ for $1 \leq i \leq k-1$ and $(u_k, u_1) \in E$. If we let $\mathcal{L} = \{\mathsf{R}\}$, where R is a binary relation symbol, then directed graphs correspond exactly to $\mathcal{L}$-structures. Show that the class of directed acyclic graphs (that is, directed graphs with no cycles) is an elementary class in this language.

2. (a) Let $\mathcal{L} = \{\mathsf{f}\}$ where f is a binary function symbol. Show that $(\mathbb{N}, +) \not\equiv (\mathbb{Z}, +)$.

   (b) $\mathcal{L} = \{\mathsf{f}\}$ where f is a binary function symbol. Define $g\colon \{1,2,3,4\}^2 \to \{1,2,3,4\}$ and $h\colon \{a,b,c,d\}^2 \to \{a,b,c,d\}$ by

g	1	2	3	4
1	4	3	1	1
2	2	2	1	2
3	1	4	1	4
4	1	3	2	3

h	a	b	c	d
a	b	b	c	b
b	a	d	d	a
c	b	a	c	a
d	d	b	c	a

   Interpret the diagrams as follows. If $m, n \in \{1,2,3,4\}$, to calculate the value of $g(m,n)$, go to row m and column n. For example, $g(1,2) = 3$. Similarly for h. Show that $(\{1,2,3,4\}, g) \not\equiv (\{a,b,c,d\}, h)$.

   (c) Let $\mathcal{L} = \{\mathsf{R}\}$ where R is a 3-ary relation symbol. Let $\mathcal{M}$ be the $\mathcal{L}$-structure where $M = \mathbb{R}$ and $\mathsf{R}^{\mathcal{M}}$ is the "betweenness relation", i.e. $\mathsf{R}^{\mathcal{M}} = \{(a,b,c) \in \mathbb{R}^3 :$ Either $a \leq b \leq c$ or $c \leq b \leq a\}$. Let $\mathcal{N}$ be the $\mathcal{L}$-structure where $N = \mathbb{R}^2$ and $\mathsf{R}^{\mathcal{N}}$ is the "collinearity relation", i.e. $\mathsf{R}^{\mathcal{N}} = \{((x_1,y_1),(x_2,y_2),(x_3,y_3)) \in (\mathbb{R}^2)^3 :$ There exists $a, b, c \in \mathbb{R}$ with either $a \neq 0$ or $b \neq 0$ such that $ax_i + by_i = c$ for all $i\}$. Show that $\mathcal{M} \not\equiv \mathcal{N}$.

3. Let $\mathcal{L} = \{\mathsf{f}\}$ where f is a binary function symbol. Let $\mathcal{M}$ be the $\mathcal{L}$-structure where $M = \{0,1\}^*$ and $\mathsf{f}^{\mathcal{M}}\colon M^2 \to M$ is concatenation (i.e. $\mathsf{f}^{\mathcal{M}}(\sigma, \tau) = \sigma\tau$).

   (a) Show that $\{\lambda\} \subseteq M$ is definable in $\mathcal{M}$.

   (b) Show that for each $n \in \mathbb{N}$, the set $\{\sigma \in M : |\sigma| = n\}$ is definable in $\mathcal{M}$.

   (c) Find all automorphisms of $\mathcal{M}$.

   (d) Show that $\{\sigma \in M : \sigma$ contains no 1's$\} = \{0\}^*$ is not definable in $\mathcal{M}$.

4. Let $\mathcal{L} = \{\mathsf{f}\}$ where f is a binary function symbol. Let $\mathcal{M}$ be the $\mathcal{L}$-structure where $M = \mathbb{N}$ and $\mathsf{f}^{\mathcal{M}}\colon M^2 \to M$ is multiplication (i.e. $\mathsf{f}^{\mathcal{M}}(m,n) = m \cdot n$).

   (a) Show that $\{0\} \subseteq M$ is definable in $\mathcal{M}$.

   (b) Show that $\{1\} \subseteq M$ is definable in $\mathcal{M}$.

   (c) Show that $\{p \in M : p$ is prime$\}$ is definable in $\mathcal{M}$.

   (d) Find all automorphisms of $\mathcal{M}$.

   (e) Show that $\{n\} \subseteq M$ is not definable in $\mathcal{M}$ whenever $n \geq 2$.

   (f) Show that $\{(k,m,n) \in M^3 : k + m = n\}$ is not definable in $\mathcal{M}$.

5. Let $\mathcal{L} = \{\mathsf{e}, \mathsf{f}\}$ be the restricted group theory language. Let $\mathcal{M}$ be the symmetric group S_4, and let $X \subseteq M$ be the set of all transpositions (i.e. 2-cycles) in S_4. Show that X is definable in $\mathcal{M}$. If you give an explicit formula, you should explain why it works.

6. Let $\mathcal{L} = \{\mathsf{P}\}$ where P is a unary relation symbol. Let $\mathcal{M}$ be the $\mathcal{L}$-structure where $M = \mathbb{Q}$ and $\mathsf{P}^{\mathcal{M}} = \{q \in \mathbb{Q} : q > 0\}$. Let $\mathcal{N}$ be the $\mathcal{L}$-structure where $N = \mathbb{N}$ and $\mathsf{P}^{\mathcal{N}} = \{2n : n \in \mathbb{N}\}$. Show that $\mathcal{M} \cong \mathcal{N}$.

7. (a) Let $\mathcal{L} = \{\mathsf{P}\}$ where P is a unary relation symbol and let $\mathcal{M}$ be a finite $\mathcal{L}$-structure. Show that there exists $\sigma \in Sent_{\mathcal{L}}$ such that for all $\mathcal{L}$-structures $\mathcal{N}$, we have

$$\mathcal{N} \vDash \sigma \text{ if and only if } \mathcal{M} \cong \mathcal{N}.$$

   (b) Solve part (a) in the case of $\mathcal{L} = \{\mathsf{R}\}$, where R is a binary relation symbol.

   Note: This problem generalizes to any finite language $\mathcal{L}$. In particular, if $\mathcal{L}$ is a finite language and $\mathcal{M}$ is a finite $\mathcal{L}$-structure, then for any $\mathcal{L}$-structure $\mathcal{N}$, we have $\mathcal{M} \equiv \mathcal{N}$ if and only if $\mathcal{M} \cong \mathcal{N}$. In other words, if $\mathcal{M}$ is finite, then being elementarily equivalent to $\mathcal{M}$ is the same thing as being isomorphic to $\mathcal{M}$. In contrast, we will show that this is *never* the case for an infinite $\mathcal{L}$-structure $\mathcal{M}$.

8. Let $\varphi \in Form_{\mathcal{L}}$, $t \in Term_{\mathcal{L}}$, and $\mathsf{x} \in Var$. Show that if $\mathsf{x} \notin OccurVar(t)$, then $\mathsf{x} \notin FreeVar(\varphi^t_{\mathsf{x}})$.

9. Let $\mathcal{M}$ be an $\mathcal{L}$-structure, let $s\colon Var \to M$, let $\varphi \in Form_{\mathcal{L}}$, and let $\mathsf{x}, \mathsf{y} \in Var$. If $\mathsf{y} \notin OccurVar(\varphi)$, then $ValidSubst^{\mathsf{y}}_{\mathsf{x}}(\varphi) = 1$ and

$$(\mathcal{M}, s) \vDash \exists \mathsf{x}\varphi \text{ if and only if } (\mathcal{M}, s) \vDash \exists \mathsf{y}\varphi^{\mathsf{y}}_{\mathsf{x}}.$$

10. Let $\mathcal{M}$ be an $\mathcal{L}$-structure, let $\alpha\colon Var \to Term_{\mathcal{L}}$, and let $\mathsf{x} \in Var$. Assume that $\mathsf{x} \notin OccurVar(\alpha(\mathsf{z}))$ for all $\mathsf{z} \in Var$ with $\mathsf{z} \neq \mathsf{x}$. Show that for all $\varphi \in Form_{\mathcal{L}}$ with $\mathsf{x} \notin FreeVar(\varphi)$, we have $\mathsf{x} \notin FreeVar(\varphi^{\alpha})$.

11. Let $\mathcal{M}$ be an $\mathcal{L}$-structure, let $\alpha\colon Var \to Term_{\mathcal{L}}$, and let $\mathsf{x} \in Var$. Assume that $\mathsf{x} \notin OccurVar(\alpha(\mathsf{z}))$ for all $\mathsf{z} \in Var$.

   (a) Show that for all $u \in Term_{\mathcal{L}}$, we have $\mathsf{x} \notin OccurVar(u^{\alpha})$.

   (b) Show that for all $\varphi \in Form_{\mathcal{L}}$, we have $\mathsf{x} \notin FreeVar(\varphi^{\alpha})$.

12. Let $\mathcal{L}$ be a language and let $\alpha\colon Var \to Term_{\mathcal{L}}$. Assume that whenever $\mathsf{y} \in OccurVar(\alpha(\mathsf{x}))$ and $\mathsf{y} \neq \mathsf{x}$, we have $\mathsf{y} \notin BoundVar(\varphi)$. Show that $ValidSubst^{\alpha}(\varphi) = 1$.

13. (**) Let $\mathcal{L} = \{0, 1, +, -, \cdot\}$ be the language of rings. Show that $\mathbb{R} \subseteq \mathbb{C}$ is not definable in $(\mathbb{C}, 0, 1, +, -, \cdot)$.

Chapter 5

Theories and Models

5.1 Semantic Implication

In propositional logic, we needed a truth assignment $M\colon P \to \{0,1\}$ in order to assign true/false values to each formula $\varphi \in Form_P$. For first-order logic, we need an $\mathcal{L}$-structure $\mathcal{M}$ and a variable assignment $s\colon Var \to M$ in order to assign true/false values to each formula $\varphi \in Form_{\mathcal{L}}$. Since these pairs $(\mathcal{M}, s)$ now provide the context that truth assignments M did in propositional logic, we can now define the first-order version of semantic implication. We start with the following definition.

Definition 5.1.1. *Let $\mathcal{L}$ be a language and let $\Gamma \subseteq Form_{\mathcal{L}}$. A* model of Γ *is a pair $(\mathcal{M}, s)$, where $\mathcal{M}$ is an $\mathcal{L}$-structure and $s\colon Var \to M$ is a variable assignment, such that $(\mathcal{M}, s) \vDash \gamma$ for all $\gamma \in \Gamma$.*

Notice that this use of the word *model* matches up with the symbolism $Mod(\Sigma)$ from Definition 4.2.9. In that setting, we had a set Σ of *sentences*, and we were looking at all $\mathcal{L}$-structures that made all of the sentences in Σ true. In other words, we were looking at the class of all models of Σ. Since sentences do not have any free variables, we did not need to worry about the variable assignment in that case.

Definition 5.1.2. *Let $\mathcal{L}$ be a language. Let $\Gamma \subseteq Form_{\mathcal{L}}$ and let $\varphi \in Form_{\mathcal{L}}$. We write $\Gamma \vDash \varphi$ to mean that whenever $(\mathcal{M}, s)$ is a model of Γ, we have that $(\mathcal{M}, s) \vDash \varphi$. We pronounce $\Gamma \vDash \varphi$ as* Γ semantically implies φ.

As mentioned after Notation 4.2.7, be very careful to distinguish between $(\mathcal{M}, s) \vDash \varphi$ and $\Gamma \vDash \varphi$. When we have a structure (together with a variable assignment) on the left, then $\vDash$ represents semantic truth. If we instead have a set of formulas on the left, then $\vDash$ represents semantic implication. Moreover the latter definition of semantic implication is defined in terms of the former definition of semantic truth.

For example, let $\mathcal{L} = \{\mathsf{f}, \mathsf{g}\}$ where f and g are unary function symbols. We claim that

$$\{\forall \mathsf{x}(\mathsf{fgx} = \mathsf{x}), \forall \mathsf{x}(\mathsf{gfx} = \mathsf{x})\} \vDash \forall \mathsf{y}\exists \mathsf{x}(\mathsf{fx} = \mathsf{y}) \wedge \forall \mathsf{y}\exists \mathsf{x}(\mathsf{gx} = \mathsf{y}).$$

To see this, let $(\mathcal{M}, s)$ be an arbitrary model of $\{\forall \mathsf{x}(\mathsf{fgx} = \mathsf{x}), \forall \mathsf{x}(\mathsf{gfx} = \mathsf{x})\}$. We then have that $\mathsf{f}^{\mathcal{M}}\colon M \to M$ and $\mathsf{g}^{\mathcal{M}}\colon M \to M$ are inverses of each other. It follows that both of the functions $\mathsf{f}^{\mathcal{M}}$ and $\mathsf{g}^{\mathcal{M}}$ are bijective, and so in particular both are surjective. Therefore, $(\mathcal{M}, s) \vDash \forall \mathsf{y}\exists \mathsf{x}(\mathsf{fx} = \mathsf{y}) \wedge \forall \mathsf{y}\exists \mathsf{x}(\mathsf{gx} = \mathsf{y})$. Notice that the variable assignment s played no role in any of our reasoning here, because all of the formulas involved are sentences.

For another example, let $\mathcal{L} = \{\mathsf{f}, \mathsf{R}\}$ where f is a unary function symbol and R is a unary relation symbol. We claim that

$$\{\forall \mathsf{y}(\mathsf{Ry} \to \mathsf{fy} = \mathsf{y}), \mathsf{Rx}\} \vDash \mathsf{fx} = \mathsf{x}.$$

To see this formally, let $(\mathcal{M}, s)$ be an arbitrary model of $\{\forall \mathsf{y}(\mathsf{Ry} \to \mathsf{fy} = \mathsf{y}), \mathsf{Rx}\}$. Let $a = s(\mathsf{x})$. Since $(\mathcal{M}, s) \vDash \mathsf{Rx}$, we have $s(\mathsf{x}) \in \mathsf{R}^{\mathcal{M}}$, which means that $a \in \mathsf{R}^{\mathcal{M}}$. Now we also have $(\mathcal{M}, s) \vDash \forall \mathsf{y}(\mathsf{Ry} \to \mathsf{fy} = \mathsf{y})$, so in particular we know that $(\mathcal{M}, s[\mathsf{y} \Rightarrow a]) \vDash \mathsf{Ry} \to \mathsf{fy} = \mathsf{y}$. Since $(\mathcal{M}, s[\mathsf{y} \Rightarrow a]) \vDash \mathsf{Ry}$, we conclude that $(\mathcal{M}, s[\mathsf{y} \Rightarrow a]) \vDash \mathsf{fy} = \mathsf{y}$, hence $\mathsf{f}^{\mathcal{M}}(a) = a$. Since $s(\mathsf{x}) = a$, it follows that $(\mathcal{M}, s) \vDash \mathsf{fx} = \mathsf{x}$.

However, we claim that

$$\{\forall \mathsf{y}(\mathsf{Ry} \to \mathsf{fy} = \mathsf{y}), \mathsf{Rx}\} \nvDash \mathsf{fy} = \mathsf{y}.$$

To see this, it suffices to give a model $(\mathcal{M}, s)$ of $\{\forall \mathsf{y}(\mathsf{Ry} \to \mathsf{fy} = \mathsf{y}), \mathsf{Rx}\}$ such that $(\mathcal{M}, s) \nvDash \mathsf{fy} = \mathsf{y}$. Let $\mathcal{M}$ be the structure with $M = \{1, 2\}$, with $\mathsf{R}^{\mathcal{M}} = \{1\}$, and with $\mathsf{f}^{\mathcal{M}}$ equal to the function with $\mathsf{f}^{\mathcal{M}}(1) = 1$ and $\mathsf{f}^{\mathcal{M}}(2) = 1$. Let $s \colon Var \to M$ be the variable assignment with

$$s(\mathsf{z}) = \begin{cases} 2 & \text{if } \mathsf{z} = \mathsf{y} \\ 1 & \text{otherwise.} \end{cases}$$

It is then straightforward to check that $(\mathcal{M}, s)$ is a model of $\{\forall \mathsf{y}(\mathsf{Ry} \to \mathsf{fy} = \mathsf{y}), \mathsf{Rx}\}$, but $(\mathcal{M}, s) \nvDash \mathsf{fy} = \mathsf{y}$.

Now consider the group theory language $\mathcal{L} = \{\mathsf{e}, *, -1\}$. The group axioms can be written as $\mathcal{L}$-sentences:

$$\begin{aligned} \sigma_1 &: \forall \mathsf{x} \forall \mathsf{y} \forall \mathsf{z}((\mathsf{x} * \mathsf{y}) * \mathsf{z} = \mathsf{x} * (\mathsf{y} * \mathsf{z})) \\ \sigma_2 &: \forall \mathsf{x}((\mathsf{x} * \mathsf{e} = \mathsf{x}) \wedge (\mathsf{e} * \mathsf{x} = \mathsf{x})) \\ \sigma_3 &: \forall \mathsf{x}((\mathsf{x} * \mathsf{x}^{-1} = \mathsf{e}) \wedge (\mathsf{x}^{-1} * \mathsf{x} = \mathsf{e})). \end{aligned}$$

We then have

$$\{\sigma_1, \sigma_2, \sigma_3\} \vDash \forall \mathsf{x} \forall \mathsf{y} \forall \mathsf{z}((\mathsf{x} * \mathsf{y} = \mathsf{x} * \mathsf{z} \to \mathsf{y} = \mathsf{z})$$

by simple properties of groups (i.e. if $a, b, c \in G$ and $ab = ac$, then $b = c$). However, notice that we have both

$$\{\sigma_1, \sigma_2, \sigma_3\} \nvDash \forall \mathsf{x} \forall \mathsf{y}(\mathsf{x} * \mathsf{y} = \mathsf{y} * \mathsf{x})$$

and

$$\{\sigma_1, \sigma_2, \sigma_3\} \nvDash \neg \forall \mathsf{x} \forall \mathsf{y}(\mathsf{x} * \mathsf{y} = \mathsf{y} * \mathsf{x})$$

because there exist both nonabelian groups and abelian groups. In particular, given Γ and φ, it is possible that both $\Gamma \nvDash \varphi$ and $\Gamma \nvDash \neg\varphi$. This is a key distinction between what happens when we have a structure together with a variable assignment $(\mathcal{M}, s)$ on the left of the $\vDash$, versus a set of formulas. Recall that for every formula φ, we have that exactly one of $(\mathcal{M}, s) \vDash \varphi$ and $(\mathcal{M}, s) \vDash \neg\varphi$ is true, due to the recursive definition of $\vDash$ in a structure. Always be mindful of how to interpret $\vDash$ by looking at the type of object on the left!

Similarly, suppose that $\mathcal{L} = \{\mathsf{R}\}$, where R is a binary relation symbol. The partial ordering axioms can be written as $\mathcal{L}$-sentences:

$$\begin{aligned} \sigma_1 &: \forall \mathsf{x} \mathsf{Rxx} \\ \sigma_2 &: \forall \mathsf{x} \forall \mathsf{y}((\mathsf{Rxy} \wedge \mathsf{Ryx}) \to (\mathsf{x} = \mathsf{y})) \\ \sigma_3 &: \forall \mathsf{x} \forall \mathsf{y} \forall \mathsf{z}((\mathsf{Rxy} \wedge \mathsf{Ryz}) \to \mathsf{Rxz}). \end{aligned}$$

We have both

$$\{\sigma_1, \sigma_2, \sigma_3\} \nvDash \forall \mathsf{x} \forall \mathsf{y}(\mathsf{Rxy} \vee \mathsf{Ryx})$$

and

$$\{\sigma_1, \sigma_2, \sigma_3\} \nvDash \neg \forall \mathsf{x} \forall \mathsf{y}(\mathsf{Rxy} \vee \mathsf{Ryx})$$

because some partial ordering are linear orderings and some are not.

For a related cautionary example, consider the $\vee$ connective. Recall that given a structure $\mathcal{M}$, a variable assignment $s \colon Var \to M$, and $\varphi, \psi \in Form_{\mathcal{L}}$, we have that $(\mathcal{M}, s) \vDash \varphi \vee \psi$ if and only if either $(\mathcal{M}, s) \vDash \varphi$ or

$(\mathcal{M}, s) \vDash \psi$ by the recursive definition of truth in a structure. However, if $\Gamma \subseteq Form_{\mathcal{L}}$ and $\varphi, \psi \in Form_{\mathcal{L}}$, then in general we do *not* have that $\Gamma \vDash \varphi \vee \psi$ if and only if either $\Gamma \vDash \varphi$ or $\Gamma \vDash \psi$. The right-to-left direction is true in this case, but not the left-to-right direction. For example, if $\{\sigma_1, \sigma_2, \sigma_3\}$ is the set of group theory axioms from above, and τ is the sentence asserting that the binary operations is commutative, then $\{\sigma_1, \sigma_2, \sigma_3\} \vDash \tau \vee (\neg\tau)$, but $\{\sigma_1, \sigma_2, \sigma_3\} \nvDash \tau$ and $\{\sigma_1, \sigma_2, \sigma_3\} \nvDash \neg\tau$.

Sticking with the group theory language and axioms, notice that the formulas σ_1, σ_2, and σ_3 are all sentences. If we add a formula $\psi(\mathsf{x})$ to our assumptions, then a model of $\{\sigma_1, \sigma_2, \sigma_3, \psi\}$ will come equipped with a variable assignment s that will send x to an element of the model. For example, a model of

$$\{\sigma_1, \sigma_2, \sigma_3, \mathsf{x} * \mathsf{x} = \mathsf{x}\}$$

is a group together with an element a (the value of $s(\mathsf{x})$) such that $a^2 = a$. Given any such model $(\mathcal{M}, a)$, notice that we must have $a = e$ because we can multiply both sides of the equation $a^2 = a$ by a^{-1}. Therefore,

$$\{\sigma_1, \sigma_2, \sigma_3, \mathsf{x} * \mathsf{x} = \mathsf{x}\} \vDash \mathsf{x} = \mathsf{e}.$$

In other words, if Γ contains formulas with free variables, then a model of Γ comes equipped with specific elements of the (universe of the) model that satisfy those formulas. Moreover, if φ is a formula with some of the same free variables, then $\Gamma \vDash \varphi$ means that whenever we have a model with specific elements that satisfy the corresponding formulas in Γ, then those same elements satisfy φ.

Now consider the situation where $\Gamma \vDash \varphi$, but φ contains a free variable that does not occur free in Γ. For example, with the same group theory axioms, we have

$$\{\sigma_1, \sigma_2, \sigma_3\} \nvDash \mathsf{x} = \mathsf{e},$$

because there exists a group G together with an element a such that $a \neq e$. Notice that since x does not occur free in $\{\sigma_1, \sigma_2, \sigma_3\}$, we made no assumptions about what $s(\mathsf{x})$ must satisfy in a model. In contrast, we do have

$$\{\sigma_1, \sigma_2, \sigma_3\} \vDash (\neg(\mathsf{x} = \mathsf{e}) \wedge (\mathsf{x} * \mathsf{x} = \mathsf{e})) \rightarrow \neg(\mathsf{x} * (\mathsf{x} * \mathsf{x}) = \mathsf{e})$$

because given any group G and any $a \in G$ with order 2, we have $a^3 \neq e$. Again, since x does not appear as a free variable in $\{\sigma_1, \sigma_2, \sigma_3\}$, we are not making any assumptions about x, so we have to consider all possible values of $s(\mathsf{x})$ in any model. Thinking through the consequences, we arrive at the following simple result.

Proposition 5.1.3. *Let $\mathcal{L}$ be a language, let $\Gamma \subseteq Form_{\mathcal{L}}$, and let $\varphi \in Form_{\mathcal{L}}$. Let $\mathsf{x} \in Var$ be such that $\mathsf{x} \notin FreeVar(\Gamma)$, i.e. such that $\mathsf{x} \notin FreeVar(\gamma)$ for all $\gamma \in \Gamma$. We then have $\Gamma \vDash \varphi$ if and only if $\Gamma \vDash \forall\mathsf{x}\varphi$*

Proof. Assume first that $\Gamma \vDash \varphi$. We show that $\Gamma \vDash \forall\mathsf{x}\varphi$. Let $(\mathcal{M}, s)$ be an arbitrary model of Γ, and let $a \in M$ be arbitrary. Since $(\mathcal{M}, s)$ is a model of Γ and $\mathsf{x} \notin FreeVar(\Gamma)$, it follows from Proposition 4.2.6 that $(\mathcal{M}, s[\mathsf{x} \Rightarrow a])$ is a model of Γ. Since $\Gamma \vDash \varphi$, we conclude that $(\mathcal{M}, s[\mathsf{x} \Rightarrow a]) \vDash \varphi$. We have shown that $(\mathcal{M}, s[\mathsf{x} \Rightarrow a]) \vDash \varphi$ whenever $(\mathcal{M}, s)$ is a model of Γ and $a \in M$, so it follows that $(\mathcal{M}, s) \vDash \forall\mathsf{x}\varphi$.

For the converse, assume that $\Gamma \vDash \forall\mathsf{x}\varphi$. We show that $\Gamma \vDash \varphi$. Let $(\mathcal{M}, s)$ be an arbitrary model of Γ. Since $\Gamma \vDash \forall\mathsf{x}\varphi$, it follows that $(\mathcal{M}, s) \vDash \forall\mathsf{x}\varphi$. By definition, we conclude that $(\mathcal{M}, s[\mathsf{x} \Rightarrow a]) \vDash \varphi$ for all $a \in M$, so in particular it follows that $(\mathcal{M}, s[\mathsf{x} \Rightarrow s(\mathsf{x})]) \vDash \varphi$. Since $s[\mathsf{x} \Rightarrow s(\mathsf{x})] = s$, we conclude that $(\mathcal{M}, s) \vDash \varphi$. We have shown that $(\mathcal{M}, s)$ is a model of φ whenever $(\mathcal{M}, s)$ is a model of Γ, so $\Gamma \vDash \varphi$. □

By iterating this result, we obtain the following corollary.

Corollary 5.1.4. *Let $\mathcal{L}$ be a language, let $\Gamma \subseteq Form_{\mathcal{L}}$, and let $\varphi \in Form_{\mathcal{L}}$. Let $\mathsf{x}_1, \mathsf{x}_2, \ldots, \mathsf{x}_n \in Var$ be variables such that $\mathsf{x}_i \notin FreeVar(\Gamma)$ for all i. We then have $\Gamma \vDash \varphi$ if and only if $\Gamma \vDash \forall\mathsf{x}_1\forall\mathsf{x}_2 \ldots \forall\mathsf{x}_n\varphi$.*

As a special case, notice that if Γ is a set of sentences, then we lose no information by only considering sentences on the right of $\vDash$, because we can always replace such a formal by its *universal closure*, i.e. the formula obtained by universally quantifying all of the free variables.

What happens if we consider constant symbols that do not occur in Γ? Intuitively, since Γ places no restriction on the interpretation of such a symbol, it should act similarly to a variable that does not occur free in Γ. In other words, in this situation, $\Gamma \vDash \varphi$ should be true if and only if $\Gamma \vDash \varphi^{\mathsf{c}}_{\mathsf{x}}$. Now in order to prove this result carefully, we should first formally define what it means to say that "c does not occur in a formula ψ", but it is straightforward to define the set of terms that occur in a formula ψ recursively by following a hybrid between the definition of $SubForm$ (in Definition 3.1.16) and $OccurVar$ (in Definition 4.1.15). In the proof, we will also make use of the fact that if c does not occur in a formula ψ, then changing the value $\mathsf{c}^{\mathcal{M}}$ in a structure does not affect whether $(\mathcal{M}, s) \vDash \psi$. This intuitively clear result is analogous to Proposition 4.2.6, and follows from a completely straightforward induction, so will be omitted.

Proposition 5.1.5. *Let $\mathcal{L}$ be a language, let $\Gamma \subseteq Form_{\mathcal{L}}$, and let $\varphi \in Form_{\mathcal{L}}$. Let c be a constant that does not occur in any formula of $\Gamma \cup \{\varphi\}$, and let $\mathsf{x} \in Var$ be such that $\mathsf{x} \notin FreeVar(\Gamma)$. We then have $\Gamma \vDash \varphi$ if and only if $\Gamma \vDash \varphi^{\mathsf{c}}_{\mathsf{x}}$.*

Proof. Assume first that $\Gamma \vDash \varphi$. We show that $\Gamma \vDash \varphi^{\mathsf{c}}_{\mathsf{x}}$. Let $(\mathcal{M}, s)$ be an arbitrary model of Γ. Since $\mathsf{x} \notin FreeVar(\Gamma)$, it follows from Proposition 4.2.6 that $(\mathcal{M}, s[\mathsf{x} \Rightarrow \mathsf{c}^{\mathcal{M}}])$ is a model of Γ. Using the fact that $\Gamma \vDash \varphi$, we conclude that $(\mathcal{M}, s[\mathsf{x} \Rightarrow \mathsf{c}^{\mathcal{M}}]) \vDash \varphi$. Now $\mathsf{c}^{\mathcal{M}} = \overline{s}(\mathsf{c})$ by definition, and $ValidSubst^{\mathsf{c}}_{\mathsf{x}}(\varphi) = 1$ by Proposition 4.6.6 as $OccurVar(\mathsf{c}) = \emptyset$. Therefore, by the Substitution Theorem (Theorem 4.6.7), it follows that $(\mathcal{M}, s) \vDash \varphi^{\mathsf{c}}_{\mathsf{x}}$. Since $(\mathcal{M}, s)$ was an arbitrary model of Γ, we conclude that $\Gamma \vDash \varphi^{\mathsf{c}}_{\mathsf{x}}$.

For the converse, assume that $\Gamma \vDash \varphi^{\mathsf{c}}_{\mathsf{x}}$. We show that $\Gamma \vDash \varphi$. Let $(\mathcal{M}, s)$ be an arbitrary model of Γ. Define a new $\mathcal{L}$-structure $\mathcal{N}$ by letting it have the same underlying set and interpretations as $\mathcal{M}$, except for setting $\mathsf{c}^{\mathcal{N}} = s(\mathsf{x})$. Notice $(\mathcal{N}, s)$ is also a model of Γ because c does not occur in Γ. Since $\Gamma \vDash \varphi^{\mathsf{c}}_{\mathsf{x}}$, it follows that $(\mathcal{N}, s) \vDash \varphi^{\mathsf{c}}_{\mathsf{x}}$. Using the Substitution Theorem (Theorem 4.6.7), we conclude that $(\mathcal{N}, s[\mathsf{x} \Rightarrow \overline{s}(\mathsf{c})]) \vDash \varphi$. Now $\overline{s}(\mathsf{c}) = \mathsf{c}^{\mathcal{N}} = s(\mathsf{x})$, so $s[\mathsf{x} \Rightarrow \overline{s}(\mathsf{c})] = s$, and hence $(\mathcal{N}, s) \vDash \varphi$. As c also does not occur in φ, we conclude that $(\mathcal{M}, s) \vDash \varphi$. Since $(\mathcal{M}, s)$ was an arbitrary model of Γ, we conclude that $\Gamma \vDash \varphi$. □

By combining Proposition 5.1.3 and Proposition 5.1.5, we arrive at the following result, which will play an important role in the next couple of chapters.

Corollary 5.1.6. *Let $\mathcal{L}$ be a language, let $\Gamma \subseteq Form_{\mathcal{L}}$, and let $\varphi \in Form_{\mathcal{L}}$. Let c be a constant that does not occur in any formula of $\Gamma \cup \{\varphi\}$, and let $\mathsf{x} \in Var$ be such that $\mathsf{x} \notin FreeVar(\Gamma)$. The following are equivalent:*

1. $\Gamma \vDash \varphi$.
2. $\Gamma \vDash \varphi^{\mathsf{c}}_{\mathsf{x}}$.
3. $\Gamma \vDash \forall \mathsf{x} \varphi$.

The next result is the multiple substitution variant of the previous corollary. Since we are only substituting constants (which do not contain variables), it turns out that simultaneous substitution is the same as iterated substitution. If the reader accepts this fact, then the result follows by simply iterating Corollary 5.1.6. However, one can also prove it by following the outline of Proposition 5.1.5, but using the Multiple Substitution Theorem together with Exercise 12 in Chapter 4.

Proposition 5.1.7. *Let $\mathcal{L}$ be a language, let $\Gamma \subseteq Form_{\mathcal{L}}$, and let $\varphi \in Form_{\mathcal{L}}$. Let $\mathsf{c}_1, \ldots, \mathsf{c}_n$ be distinct constants that do not occur in any formula of $\Gamma \cup \{\varphi\}$, and let $\mathsf{x}_1, \ldots, \mathsf{x}_n \in Var$ be distinct variables such that $\mathsf{x}_i \notin FreeVar(\Gamma)$ for all i. Define $\alpha \colon Var \to Term_{\mathcal{L}}$ by letting*

$$\alpha(\mathsf{z}) = \begin{cases} \mathsf{c}_i & \text{if } \mathsf{z} = \mathsf{x}_i \\ \mathsf{z} & \text{otherwise.} \end{cases}$$

The following are equivalent:

1. $\Gamma \vDash \varphi$.

2. $\Gamma \vDash \varphi^{\alpha}$.

3. $\Gamma \vDash \forall \mathsf{x}_1 \forall \mathsf{x}_2 \ldots \forall \mathsf{x}_n \varphi$.

We can also define satisfiability in the first-order context, in analogy with how we defined it in propositional logic.

Definition 5.1.8. *Let $\mathcal{L}$ be a language and let $\Gamma \subseteq Form_{\mathcal{L}}$. We say that Γ is* satisfiable *if there exists a model of Γ. Otherwise, we say that Γ is* unsatisfiable.

Using our hard work on elementary substructures, we immediately obtain the following result.

Theorem 5.1.9 (Countable Lowenheim-Skolem Theorem)**.** *Suppose that $\mathcal{L}$ is countable and that $\Gamma \subseteq Form_{\mathcal{L}}$ is satisfiable. There exists a countable model $(\mathcal{M}, s)$ of Γ.*

Proof. Since Γ is satisfiable, we may fix a model $(\mathcal{N}, s)$ of Γ. Let $X = \text{range}(s) \subseteq N$ and notice that X is countable. By the Countable Lowenheim-Skolem-Tarski Theorem, there exists a countable elementary substructure $\mathcal{M} \preceq \mathcal{N}$ such that $X \subseteq M$. Notice that s is also a variable assignment on M. Now for any $\gamma \in \Gamma$, we have that $(\mathcal{N}, s) \vDash \gamma$ because $(\mathcal{N}, s)$ is a model of Γ, hence $(\mathcal{M}, s) \vDash \gamma$ because $\mathcal{M} \preceq \mathcal{N}$. It follows that $(\mathcal{M}, s)$ is a model of Γ. □

As in the propositional logic, we have the same fundamental connection between semantic implication and satisfiability.

Proposition 5.1.10. *Let $\mathcal{L}$ be a language, let $\Gamma \subseteq Form_{\mathcal{L}}$, and let $\varphi \in Form_{\mathcal{L}}$. The following are equivalent.*

1. $\Gamma \vDash \varphi$.

2. $\Gamma \cup \{\neg\varphi\}$ *is unsatisfiable.*

Proof. We prove the contrapositive of each direction. Suppose first that (1) is false, i.e. that $\Gamma \nvDash \varphi$. By definition, we can fix an $\mathcal{L}$-structure $\mathcal{M}$ and variable assignment $s\colon Var \to M$ such that $(\mathcal{M}, s) \vDash \gamma$ for all $\gamma \in \Gamma$, but $(\mathcal{M}, s) \nvDash \varphi$. By the recursive definition of $\vDash$, we then have $(\mathcal{M}, s) \vDash \neg\varphi$. Therefore, $(\mathcal{M}, s)$ is a model of $\Gamma \cup \{\neg\varphi\}$, so $\Gamma \cup \{\neg\varphi\}$ is satisfiable. Hence, (2) is false.

Suppose now that (2) is false, i.e. that $\Gamma \cup \{\neg\varphi\}$ is satisfiable. By definition, we can fix a model $(\mathcal{M}, s)$ of $\Gamma \cup \{\neg\varphi\}$. We then have $(\mathcal{M}, s) \vDash \gamma$ for all $\gamma \in \Gamma$, and also that $(\mathcal{M}, s) \vDash \neg\varphi$. By the recursive definition of $\vDash$, we have $(\mathcal{M}, s) \nvDash \varphi$. We have found a model of $(\mathcal{M}, s)$ of Γ with $(\mathcal{M}, s) \nvDash \varphi$, so $\Gamma \nvDash \varphi$. Hence, (1) is false. □

Suppose that Γ is a finite set of formulas and φ is a formula. In propositional logic, we could use truth tables, i.e. try all of the finitely many truth assignments on the variables appearing in $\Gamma \cup \{\varphi\}$, in order to determine whether $\Gamma \vDash \varphi$ Similarly, we could simply try all truth assignments to determine whether a finite set is satisfiable. Although tedious and quite slow (both take exponential time), at least there was an algorithm. In contrast, there is no obvious method that works in the first-order logic case. Intuitively, it appears that we would have to examine *all* possible $\mathcal{L}$-structures and variable assignments to determine whether $\Gamma \vDash \varphi$. Of course, there are infinitely many $\mathcal{L}$-structures. Even worse, many of these $\mathcal{L}$-structures are themselves infinite, so it's not even clear whether it's possible to check that a given pair $(\mathcal{M}, s)$ is a model of Γ. We'll have a lot more to say about these ideas in Chapter 13.

5.2 Theories

Although we will sometimes have reason to work with formulas when considering semantic implication, we will focus our attention on sentences. In that setting, we do not need to consider a variable assignment when thinking about models. Certain collections of sentences are especially fundamental.

Definition 5.2.1. *Let $\mathcal{L}$ be a language. An* $\mathcal{L}$-theory*, or simply a* theory*, is a set $T \subseteq Sent_{\mathcal{L}}$ such that whenever $\sigma \in Sent_{\mathcal{L}}$ and $T \vDash \sigma$, we have $\sigma \in T$.*

In other words, a theory is a set of sentences that is closed under semantic implication (for sentences). Note that since we are assuming that $T \subseteq Sent_{\mathcal{L}}$, the brief discussion after Corollary 5.1.4 says that we do not lose any information by only considering sentences on the right-hand side of $\vDash$.

There a couple of standard ways to build a theory. The first way is to start with an arbitrary set of sentences, and close it off under semantic implication.

Definition 5.2.2. *Let $\mathcal{L}$ be a language and let $\Sigma \subseteq Sent_{\mathcal{L}}$. We let $Cn(\Sigma) = \{\tau \in Sent_{\mathcal{L}} : \Sigma \vDash \tau\}$. We call $Cn(\Sigma)$ the set of* consequences *of Σ.*

For example, let $\mathcal{L} = \{\mathsf{e}, *, -1\}$ be the group theory language, and consider the following sentences:

$$\sigma_1 : \forall \mathsf{x} \forall \mathsf{y} \forall \mathsf{z}((\mathsf{x} * \mathsf{y}) * \mathsf{z} = \mathsf{x} * (\mathsf{y} * \mathsf{z}))$$
$$\sigma_2 : \forall \mathsf{x}(\mathsf{x} * \mathsf{e} = \mathsf{x} \wedge \mathsf{e} * \mathsf{x} = \mathsf{x})$$
$$\sigma_3 : \forall \mathsf{x}(\mathsf{x} * \mathsf{x}^{-1} = \mathsf{e} \wedge \mathsf{x}^{-1} * \mathsf{x} = \mathsf{e}).$$

The set $Grp = Cn(\{\sigma_1, \sigma_2, \sigma_3\})$ is the set of all first-order sentences that are true in *every* group.

Notice that a set $T \subseteq Sent_{\mathcal{L}}$ is a theory if and only if $Cn(T) = T$. Although $Cn(\Sigma)$ is the set of all sentences that Σ semantically implies, it is not immediately obvious that $Cn(\Sigma)$ is a theory, because there could conceivably be *new* sentences that $Cn(\Sigma)$ implies but Σ itself does not. In other words, it is not clear that $Cn(Cn(\Sigma)) = Cn(\Sigma)$. Intuitively, it seems reasonable to believe that one iteration of "closing off" is sufficient, like the situation in analysis (where the closure of the closure of a set is just the closure). Before proving that $Cn(\Sigma)$ is always indeed a theory, we establish the following simple fact.

Proposition 5.2.3. *Let $\Sigma \subseteq Sent_{\mathcal{L}}$. Given an $\mathcal{L}$-structure $\mathcal{M}$, we have that $\mathcal{M}$ is a model of Σ if and only if $\mathcal{M}$ is a model of $Cn(\Sigma)$. In other words, in the notation of Definition 4.2.9, we have $Mod(\Sigma) = Mod(Cn(\Sigma))$.*

Proof. Notice that for all $\sigma \in \Sigma$, we trivially have $\Sigma \vDash \sigma$, so $\sigma \in Cn(\Sigma)$. Therefore, $\Sigma \subseteq Cn(\Sigma)$. It follows that any model of $Cn(\Sigma)$ is a model of Σ.

Conversely, suppose that $\mathcal{M}$ is a model of Σ. Let $\tau \in Cn(\Sigma)$ be arbitrary. By definition, we have $\Sigma \vDash \tau$. Since $\mathcal{M}$ is a model of Σ, we know by definition of semantic implication that $\mathcal{M}$ is a model of τ. Since $\tau \in Cn(\Sigma)$ was arbitrary, it follows that $\mathcal{M}$ is a model of $Cn(\Sigma)$. □

Proposition 5.2.4. *For any language $\mathcal{L}$ and any $\Sigma \subseteq Sent_{\mathcal{L}}$, the set $Cn(\Sigma)$ is an $\mathcal{L}$-theory.*

Proof. Let $\tau \in Sent_{\mathcal{L}}$ be arbitrary such that $Cn(\Sigma) \vDash \tau$. We need to show that $\tau \in Cn(\Sigma)$, i.e. that $\Sigma \vDash \tau$. Let $\mathcal{M}$ be an arbitrary model of Σ. By Proposition 5.2.3, we know that $\mathcal{M}$ is a model of $Cn(\Sigma)$. Since $Cn(\Sigma) \vDash \tau$, it follows that $\mathcal{M} \vDash \tau$. Since $\mathcal{M}$ was an arbitrary model of Σ, we conclude that $\Sigma \vDash \tau$, and hence $\tau \in Cn(\Sigma)$. □

For example, the set $Grp = Cn(\{\sigma_1, \sigma_2, \sigma_3\})$ described above is a theory, naturally called the theory of groups. Similarly, we can form the theory of rings, the theory of equivalence relations, the theory of partial orderings, etc. by taking Cn of the corresponding set of axioms.

Since $Mod(\Sigma) = Mod(Cn(\Sigma))$, it follows that an elementary class of structures is simply a class that can be expressed as $Mod(T)$ for some *theory* T. We can not say the same about strong elementary classes, because if Σ is finite, then $Cn(\Sigma)$ is infinite.

Proposition 5.2.5. *Let $\Sigma_1, \Sigma_2 \subseteq Sent_{\mathcal{L}}$. The following are equivalent:*

1. *$Cn(\Sigma_1) \subseteq Cn(\Sigma_2)$.*
2. *$Mod(\Sigma_2) \subseteq Mod(\Sigma_1)$, i.e. every model of Σ_2 is a model of Σ_1.*

Therefore, we have $Cn(\Sigma_1) = Cn(\Sigma_2)$ if and only if $Mod(\Sigma_1) = Mod(\Sigma_2)$

Proof. Suppose first that $Cn(\Sigma_1) \subseteq Cn(\Sigma_2)$. Let $\mathcal{M}$ be an arbitrary model of Σ_2. Using Proposition 5.2.3, it follows that $\mathcal{M}$ is a model of $Cn(\Sigma_2)$. Since $Cn(\Sigma_1) \subseteq Cn(\Sigma_2)$, we conclude that $\mathcal{M}$ is a model of $Cn(\Sigma_1)$, and thus $\mathcal{M}$ is a model of Σ_1 by Proposition 5.2.3. Therefore, $Mod(\Sigma_2) \subseteq Mod(\Sigma_1)$.

Conversely, suppose that $Mod(\Sigma_2) \subseteq Mod(\Sigma_1)$. Let $\tau \in Cn(\Sigma_1)$ be arbitrary. We then have that $\Sigma_1 \vDash \tau$ by definition. Now given an arbitrary model $\mathcal{M}$ of Σ_2, we have that $\mathcal{M}$ is a model of Σ_1 (since $Mod(\Sigma_2) \subseteq Mod(\Sigma_1)$), so using the fact that $\Sigma_1 \vDash \tau$, we conclude that $\mathcal{M}$ is a model of τ. We have shown that any model of Σ_2 is a model of τ, so $\Sigma_2 \vDash \tau$, and hence $\tau \in Cn(\Sigma_2)$. □

In particular, if T_1 and T_2 are theories, then using the fact that $Cn(T) = T$ for all theories T, we have $T_1 = T_2$ if and only if $Mod(T_1) = Mod(T_2)$. Now we noted above that a collection of $\mathcal{L}$-structures is an elementary class if and only if it equals $Mod(T)$ for some $\mathcal{L}$-theory T. Consider the set $\mathcal{T}$ of all $\mathcal{L}$-theories. If we think of Mod as a function from $\mathcal{T}$ to the collection of elementary classes of structures (there are some set-theoretic issues here, which we will ignore until we talk about proper class later), then the previous proposition says that Mod is an order-reversing bijection between $\mathcal{T}$ and the collection of elementary classes of $\mathcal{L}$-structures.

Our second standard way to construct a theory is to take a structure $\mathcal{M}$, and consider all of the sentences that are true in that structure.

Definition 5.2.6. *Let $\mathcal{M}$ be an $\mathcal{L}$-structure. We let $Th(\mathcal{M}) = \{\tau \in Sent_{\mathcal{L}} : \mathcal{M} \vDash \tau\}$. We call $Th(\mathcal{M})$ the* theory of $\mathcal{M}$.

Proposition 5.2.7. *If $\mathcal{M}$ is an $\mathcal{L}$-structure, then $Th(\mathcal{M})$ is an $\mathcal{L}$-theory.*

Proof. Let $\sigma \in Sent_{\mathcal{L}}$ be arbitrary such that $Th(\mathcal{M}) \vDash \sigma$. Since $\mathcal{M}$ is a model of $Th(\mathcal{M})$ by definition, it follows that $\mathcal{M} \vDash \sigma$, and hence $\sigma \in Th(\mathcal{M})$. □

For example, if $\mathcal{L}$ is the group theory language, and we let $\mathcal{M}$ be the group S_5, then $Th(\mathcal{M})$ is the set of all first-order sentences that are true in the specific group S_5. Notice that $Grp \subseteq Th(\mathcal{M})$. However, this containment is strict, because the sentence asserting that there are exactly 60 elements is an element of $Th(\mathcal{M})$, but is not an element of Grp.

Let's compare the definitions of $Cn(\Sigma)$ and $Th(\mathcal{M})$. We have

$$Cn(\Sigma) = \{\tau \in Sent_{\mathcal{L}} : \Sigma \vDash \tau\}$$
$$Th(\mathcal{M}) = \{\tau \in Sent_{\mathcal{L}} : \mathcal{M} \vDash \tau\}.$$

On the face of it, the definitions look identical. We've simply alternated between putting a set of sentences on the left of $\vDash$ and putting a structure on the left of $\vDash$. However, remember that there is a crucial different between how we interpret $\vDash$ in these two situations. To elaborate on this, we introduce the following definition.

Definition 5.2.8. *An $\mathcal{L}$-theory Σ is* complete *if for all $\tau \in Sent_{\mathcal{L}}$, either $\tau \in \Sigma$ or $\neg\tau \in \Sigma$.*

Proposition 5.2.9. *If $\mathcal{M}$ is an $\mathcal{L}$-structure, then $Th(\mathcal{M})$ is a complete $\mathcal{L}$-theory.*

Proof. We've already seen in Proposition 5.2.7 that $Th(\mathcal{M})$ is a theory. Let $\tau \in Sent_{\mathcal{L}}$ be arbitrary. If $\mathcal{M} \vDash \tau$, we then have that $\tau \in Th(\mathcal{M})$. Otherwise, we have $\mathcal{M} \nvDash \tau$, so $\mathcal{M} \vDash \neg\tau$ (by the recursive definition of $\vDash$ in a structure), and hence $\neg\tau \in Th(\mathcal{M})$. □

As mentioned above, if Σ is a set of sentences and τ is a sentence, then it is possible that both $\Sigma \nvdash \tau$ and $\Sigma \nvdash \neg\tau$ are true. In particular, the set $Cn(\Sigma)$ may not be a complete theory. For example, Grp is not complete because it neither contains $\forall \mathsf{x}(\mathsf{x} * \mathsf{x} = \mathsf{e})$ nor its negation (because there are groups in which all nonidentity elements have order 2, but there are also groups where this is not the case).

For another example, let $\mathcal{L} = \{\mathsf{R}\}$ where R is a binary relation symbol. Consider the following sentences:

$$\sigma_1 : \forall \mathsf{x} \neg \mathsf{Rxx}$$
$$\sigma_2 : \forall \mathsf{x} \forall \mathsf{y} \forall \mathsf{z}((\mathsf{Rxy} \wedge \mathsf{Ryz}) \rightarrow \mathsf{Rxz})$$
$$\sigma_3 : \forall \mathsf{x} \forall \mathsf{y}(\mathsf{x} = \mathsf{y} \vee \mathsf{Rxy} \vee \mathsf{Ryx}).$$

The theory $LO = Cn(\{\sigma_1, \sigma_2, \sigma_3\})$ is called the theory of *(strict) linear orderings*. LO is not complete because it neither contains $\exists \mathsf{y} \forall \mathsf{x}(\mathsf{x} = \mathsf{y} \vee \mathsf{Rxy})$ nor its negation, since there are linear ordering with greatest elements and linear orderings without greatest elements.

There is a third natural way to create a theory that generalizes the idea of taking $Th(\mathcal{M})$. Instead of looking at the set of sentences that are true in one structure $\mathcal{M}$, suppose that we have a class $\mathcal{K}$ of $\mathcal{L}$-structures, and we look at the set of sentences that are true in every element of $\mathcal{K}$.

Definition 5.2.10. *Given a class $\mathcal{K}$ of $\mathcal{L}$-structures, we let $Th(\mathcal{K}) = \{\tau \in Sent_{\mathcal{L}} : \mathcal{M} \vDash \tau$ for all $\mathcal{M} \in \mathcal{K}\}$.*

Notice that this notation does slightly clash with our above definition of $Th(\mathcal{M})$ in that we should have originally written $Th(\{\mathcal{M}\})$ to be consistent. However, we will continue to slightly abuse the notation.

Proposition 5.2.11. *If $\mathcal{K}$ is a class of $\mathcal{L}$-structures, then $Th(\mathcal{K})$ is an $\mathcal{L}$-theory.*

Proof. Let $\mathcal{K}$ is a class of $\mathcal{L}$-structures. Let $\tau \in Sent_{\mathcal{L}}$ be arbitrary with $Th(\mathcal{K}) \vDash \tau$. Given any $\mathcal{M} \in \mathcal{K}$, we trivially have that $\mathcal{M}$ is a model of $Th(\mathcal{K})$, so since $Th(\mathcal{K}) \vDash \tau$, we conclude that $\mathcal{M} \vDash \tau$. We have shown that $\mathcal{M} \vDash \tau$ whenever $\mathcal{M} \in \mathcal{K}$, so we conclude that $\tau \in Th(\mathcal{K})$. Therefore, $Th(\mathcal{K})$ is a theory. □

For example, in the language $\mathcal{L}$ of group theory, if we let $\mathcal{K}$ be the class of all $\mathcal{L}$-structures that are groups, then $Th(\mathcal{K})$ is another way to define the theory of groups. Also, if we let $\mathcal{M}_1$ be the symmetric group S_5, and let $\mathcal{M}_2$ be the dihedral group D_6 of the hexagon, then we can form $Th(\{\mathcal{M}_1, \mathcal{M}_2\})$ of sentences that are true in both of these groups. For example, the sentence asserting that the group is not abelian is in $Th(\{\mathcal{M}_1, \mathcal{M}_2\})$, as is the sentence asserting that all elements have order at most 6. However, in contrast to the situation of the theory of just one structure, notice that $Th(\{\mathcal{M}_1, \mathcal{M}_2\}$ is not a complete theory. For example, if τ is the sentence asserting that there are at least 15 distinct elements, then we have both $\tau \notin Th(\{\mathcal{M}_1, \mathcal{M}_2\})$ and also $\neg\tau \notin Th(\{\mathcal{M}_1, \mathcal{M}_2\})$.

Let $\mathcal{L}$ be a language. Notice that Mod gives us a way to take a set of sentences and turn it into a class of structures, the output of which will always be an elementary class. In the other direction, Th takes a class of structures and produces a set of sentences, the output of which will always be a theory. Suppose that we let $Struct_{\mathcal{L}}$ be the collection of all $\mathcal{L}$-structures. As mentioned above, there are some serious set-theoretic difficulties in doing this, but let's ignore them for the moment. If we pretend that this issue does not exist, we would then have that $Mod\colon \mathcal{P}(Sent_{\mathcal{L}}) \rightarrow \mathcal{P}(Struct_{\mathcal{L}})$ and $Th\colon \mathcal{P}(Struct_{\mathcal{L}}) \rightarrow \mathcal{P}(Sent_{\mathcal{L}})$. If we think of the "sets" $\mathcal{P}(Sent_{\mathcal{L}})$ and $\mathcal{P}(Struct_{\mathcal{L}})$ as partially ordered by $\subseteq$ then the maps Mod and Th form an (antitone) Galois connection. Under this Galois connection, the closed elements of $\mathcal{P}(Sent_{\mathcal{L}})$ are the theories, and the closed elements of $\mathcal{P}(Struct_{\mathcal{L}})$ are the elementary classes, which provides a general perspective on the statement above that Mod maps the set of theories bijectively onto the collection of elementary classes.

5.3 Counting Models of Theories

Given a theory T and an $n \in \mathbb{N}^+$, we want to count the number of models of T of cardinality n up to isomorphism. There are some technical set-theoretic difficulties here which will be elaborated upon later, but the key fact that limits the number of isomorphism classes is the following result.

Proposition 5.3.1. *Let $\mathcal{L}$ be a language and let $n \in \mathbb{N}^+$. For every $\mathcal{L}$-structure $\mathcal{M}$ with $|M| = n$, there exists an $\mathcal{L}$-structure $\mathcal{N}$ with $N = [n]$ such that $\mathcal{M} \cong \mathcal{N}$.*

Proof. Let $\mathcal{M}$ be an $\mathcal{L}$-structure with $|M| = n$. Fix a bijection $h\colon M \to [n]$. Let $\mathcal{N}$ be the $\mathcal{L}$-structure where

- $N = [n]$.
- $\mathsf{c}^{\mathcal{N}} = h(\mathsf{c}^{\mathcal{M}})$ for all $\mathsf{c} \in \mathcal{C}$.
- $\mathsf{R}^{\mathcal{N}} = \{(b_1, b_2, \ldots, b_k) \in N^k : (h^{-1}(b_1), h^{-1}(b_2), \ldots, h^{-1}(b_k)) \in \mathsf{R}^{\mathcal{N}}\}$ for all $\mathsf{R} \in \mathcal{R}_k$.
- $\mathsf{f}^{\mathcal{N}}$ is the function from N^k to N defined by $\mathsf{f}^{\mathcal{N}}(b_1, b_2, \ldots, b_k) = h(\mathsf{f}^{\mathcal{M}}(h^{-1}(b_1), h^{-1}(b_2), \ldots, h^{-1}(b_k)))$ for all $\mathsf{f} \in \mathcal{F}^k$.

It is then straightforward to check that h is an isomorphism from $\mathcal{M}$ to $\mathcal{N}$. □

Proposition 5.3.2. *If $\mathcal{L}$ is finite and $n \in \mathbb{N}^+$, then there are only finitely many $\mathcal{L}$-structures with universe $[n]$.*

Proof. Since $\mathcal{L}$ is finite and we are working with the fixed universe $[n]$, there are only a finite number of choices for each $\mathsf{c}^{\mathcal{M}}$, $\mathsf{R}^{\mathcal{M}}$, and $\mathsf{f}^{\mathcal{M}}$. □

Definition 5.3.3. *Let $\mathcal{L}$ be a finite language and let T be an $\mathcal{L}$-theory. For each $n \in \mathbb{N}^+$, let $I(T, n)$ be the number of models of T of cardinality n up to isomorphism. Formally, we consider the set of all $\mathcal{L}$-structures with universe $[n]$, and count the number of equivalence classes under the equivalence relation of isomorphism.*

For example, if Grp is the theory of groups, then $I(Grp, n)$ is a very interesting function that you study in algebra courses. For example, you show that $I(Grp, p) = 1$ for all primes p, that $I(Grp, 6) = 2$, and that $I(Grp, 8) = 5$.

Example 5.3.4. *Let $\mathcal{L} = \emptyset$ and let $T = Cn(\emptyset)$. We have $I(T, n) = 1$ for all $n \in \mathbb{N}^+$.*

Proof. First notice that for every $n \in \mathbb{N}^+$, the $\mathcal{L}$-structure $\mathcal{M}$ with universe $[n]$ is a model of T of cardinality n, so $I(T, n) \geq 1$. Now notice that if $\mathcal{M}$ and $\mathcal{N}$ are models of T of cardinality n, then any bijection $h\colon M \to N$ is an isomorphism (because $\mathcal{L} = \emptyset$), so $I(T, n) \leq 1$. It follows that $I(T, n) = 1$ for all $n \in \mathbb{N}$. □

Example 5.3.5. *$I(LO, n) = 1$ for all $n \in \mathbb{N}^+$.*

Proof. First notice that for every $n \in \mathbb{N}$, the $\mathcal{L}$-structure $\mathcal{M}$ where $M = [n]$ and $\mathsf{R}^{\mathcal{M}} = \{(k, \ell) \in [n]^2 : k < \ell\}$ is a model of LO of cardinality n, so $I(LO, n) \geq 1$. Next notice that any two linear orderings of cardinality n are isomorphic. Intuitively, this works as follows. Notice (by induction on the number of elements) that every finite linear ordering has a least element. Let $\mathcal{M}$ and $\mathcal{N}$ be two linear orderings of cardinality n. Each must have a least element, so map the least element of $\mathcal{M}$ to that of $\mathcal{N}$. Remove these elements, then map the least element remaining in $\mathcal{M}$ to the least element remaining in $\mathcal{N}$, and continue. This gives an isomorphism. Formally, you can turn this into a proof by induction on n. □

Example 5.3.6. *Let $\mathcal{L} = \{\mathsf{f}\}$ where f is a unary function symbol, and let $T = Cn(\{\forall \mathsf{x}(\mathsf{ffx} = \mathsf{x})\})$. We have $I(T, n) = \lfloor \frac{n}{2} \rfloor + 1$ for all $n \in \mathbb{N}^+$.*

Proof. Let's first analyze the finite models of T. Suppose that $\mathcal{M}$ is a model of T of cardinality n. For every $a \in M$, we then have $\mathsf{f}^{\mathcal{M}}(\mathsf{f}^{\mathcal{M}}(a)) = a$. There are now two cases. Either $\mathsf{f}^{\mathcal{M}}(a) = a$, or $\mathsf{f}^{\mathcal{M}}(a) = b \neq a$ in which case $\mathsf{f}^{\mathcal{M}}(b) = a$. Let

- $Fix_{\mathcal{M}} = \{a \in M : \mathsf{f}^{\mathcal{M}}(a) = a\}$.
- $Move_{\mathcal{M}} = \{a \in M : \mathsf{f}^{\mathcal{M}}(a) \neq a\}$.

From above, we then have that $|Move_{\mathcal{M}}|$ is even and that $|Fix_{\mathcal{M}}| + |Move_{\mathcal{M}}| = n$. Now the idea is that two models $\mathcal{M}$ and $\mathcal{N}$ of T of cardinality n are isomorphic if and only if they have the same number of fixed points, because then we can match up the fixed points and then match up the "pairings" left over to get an isomorphism. Here's a more formal argument.

We now show that if $\mathcal{M}$ and $\mathcal{N}$ are models of T of cardinality n, then $\mathcal{M} \cong \mathcal{N}$ if and only if $|Fix_{\mathcal{M}}| = |Fix_{\mathcal{N}}|$. Clearly, if $\mathcal{M} \cong \mathcal{N}$, then $|Fix_{\mathcal{M}}| = |Fix_{\mathcal{N}}|$. Suppose conversely that $|Fix_{\mathcal{M}}| = |Fix_{\mathcal{N}}|$. We then must have $|Move_{\mathcal{M}}| = |Move_{\mathcal{N}}|$. Let $X_{\mathcal{M}} \subseteq Move_{\mathcal{M}}$ be a set of cardinality $\frac{|Move_{\mathcal{M}}|}{2}$ such that $\mathsf{f}^{\mathcal{M}}(x) \neq y$ for all $x, y \in X$ (that is, we pick out one member from each pairing given by $\mathsf{f}^{\mathcal{M}}$), and let $X_{\mathcal{N}}$ be such a set for $\mathcal{N}$. Define a function $h\colon M \to N$. Fix a bijection $\alpha\colon Fix_{\mathcal{M}} \to Fix_{\mathcal{N}}$ and a bijection $\beta\colon X_{\mathcal{M}} \to X_{\mathcal{N}}$. Define h by letting $h(a) = \alpha(a)$ for all $a \in Fix_{\mathcal{M}}$, letting $h(x) = \beta(x)$ for all $x \in X_{\mathcal{M}}$, and letting $h(y) = \mathsf{f}^{\mathcal{N}}(\beta(\mathsf{f}^{\mathcal{M}}(y)))$ for all $y \in Move_{\mathcal{M}} \backslash X$. We then have that h is an isomorphism from $\mathcal{M}$ to $\mathcal{N}$.

Now we need only count how many possible values there are for $|Fix_{\mathcal{M}}|$. Let $n \in \mathbb{N}^+$. Suppose first that n is even. Since $|Move_{\mathcal{M}}|$ must be even, it follows that $|Fix_{\mathcal{M}}|$ must be even. Thus, $|Fix_{\mathcal{M}}| \in \{0, 2, 4, \ldots, n\}$, so there are $\frac{n}{2} + 1$ many possibilities, and it's easy to construct models in which each of these possibilities occurs. Suppose now that n is odd. Since $|Move_{\mathcal{M}}|$ must be even, it follows that $|Fix_{\mathcal{M}}|$ must be odd. Thus, $|Fix_{\mathcal{M}}| \in \{1, 3, 5, \ldots, n\}$, so there are $\frac{n-1}{2} + 1$ many possibilities, and it's easy to construct models in which each of these possibilities occurs. Thus, in either case, we have $I(T, n) = \lfloor \frac{n}{2} \rfloor + 1$. □

Definition 5.3.7. *Suppose that $\mathcal{L}$ is a finite language and $\sigma \in Sent_{\mathcal{L}}$. Let*

$$Spec(\sigma) = \{n \in \mathbb{N}^+ : I(Cn(\sigma), n) > 0\}.$$

The set $Spec(\sigma)$ is called the spectrum *of σ.*

Proposition 5.3.8. *There exists a finite language $\mathcal{L}$ and a $\sigma \in Sent_{\mathcal{L}}$ such that $Spec(\sigma) = \{2n : n \in \mathbb{N}^+\}$.*

Proof. We give two separate arguments. First, let $\mathcal{L}$ be the language of group theory. Let σ be the conjunction of the group axioms with the sentence $\exists \mathsf{x}(\neg(\mathsf{x} = \mathsf{e}) \wedge \mathsf{x} * \mathsf{x} = \mathsf{e})$ expressing that there is an element of order 2. Now for every $n \in \mathbb{N}^+$, the group $\mathbb{Z}/(2n)\mathbb{Z}$ is a model of σ of cardinality $2n$ because $\overline{n}$ is an element of order 2. Thus, $\{2n : n \in \mathbb{N}^+\} \subseteq Spec(\sigma)$. Suppose now that $k \in Spec(\sigma)$, and fix a model $\mathcal{M}$ of σ with cardinality k. We then have that $\mathcal{M}$ is a group with an element of order 2, so by Lagrange's Theorem it follows that $2 \mid k$, so $k \in \{2n : n \in \mathbb{N}^+\}$. It follows that $Spec(\sigma) = \{2n : n \in \mathbb{N}^+\}$.

For a second example, let $\mathcal{L} = \{\mathsf{R}\}$ where R is a binary relation symbol. Let σ be the conjunction of the following sentences:

- $\forall \mathsf{x} \mathsf{R} \mathsf{x} \mathsf{x}$.
- $\forall \mathsf{x} \forall \mathsf{y} (\mathsf{R} \mathsf{x} \mathsf{y} \to \mathsf{R} \mathsf{y} \mathsf{x})$.
- $\forall \mathsf{x} \forall \mathsf{y} \forall \mathsf{z} ((\mathsf{R} \mathsf{x} \mathsf{y} \wedge \mathsf{R} \mathsf{y} \mathsf{z}) \to \mathsf{R} \mathsf{x} \mathsf{z})$.
- $\forall \mathsf{x} \exists \mathsf{y} (\neg(\mathsf{y} = \mathsf{x}) \wedge \mathsf{R} \mathsf{x} \mathsf{y} \wedge \forall \mathsf{z} (\mathsf{R} \mathsf{x} \mathsf{z} \to (\mathsf{z} = \mathsf{x} \vee \mathsf{z} = \mathsf{y})))$.

Notice that a model of σ is simply an equivalence relation in which every equivalence class has exactly 2 elements. It is now straightforward to show that $Spec(\sigma) = \{2n : n \in \mathbb{N}^+\}$. □

Proposition 5.3.9. *There exists a finite language $\mathcal{L}$ and a $\sigma \in Sent_{\mathcal{L}}$ such that $Spec(\sigma) = \{2^n : n \in \mathbb{N}^+\}$.*

Proof. Again, we give two separate arguments. First, let $\mathcal{L}$ be the language of group theory. Let σ be the conjunction of the group axioms with the sentences $\exists \mathsf{x} \neg(\mathsf{x} = \mathsf{e})$ and $\forall \mathsf{x}(\mathsf{x} * \mathsf{x} = \mathsf{e})$ expressing that the group is nontrivial and that there every nonidentity element has order 2. Now for every $n \in \mathbb{N}^+$, the group $(\mathbb{Z}/2\mathbb{Z})^n$ is a model of σ of cardinality 2^n. Thus, $\{2^n : n \in \mathbb{N}^+\} \subseteq Spec(\sigma)$. Suppose now that $k \in Spec(\sigma)$, and fix a model $\mathcal{M}$ of σ of cardinality k. We then have that $k > 1$ and that $\mathcal{M}$ is a group such that every nonidentity

element has order 2. Now for any prime $p \neq 2$, it is not the case that p divides k because otherwise $\mathcal{M}$ would have to have an element of order p by Cauchy's Theorem. Thus, the only prime that divides k is 2, and so $k \in \{2n : n \in \mathbb{N}^+\}$. It follows that $Spec(\sigma) = \{2^n : n \in \mathbb{N}^+\}$.

For a second example, let $\mathcal{L}$ be the language of ring theory. Let σ be the conjunction of the field axioms together with $1 + 1 = 0$. Thus, the models of σ are exactly the fields of characteristic 2. By results in algebra, there is a finite field of characteristic 2 of cardinality k if and only if k is a power of 2. □

For the theory of linear orderings LO, we saw that $I(LO, n) = 1$ for all $n \in \mathbb{N}^+$. Now the theory of linear orderings is not complete, because some linear orderings have a maximum element, and some do not. For example, every finite linear ordering has a maximum element, but $(\mathbb{N}, <)$ does not. More formally, for the sentence τ equal to

$$\exists \mathsf{x} \forall \mathsf{y}(\mathsf{Ryx} \vee \mathsf{y} = \mathsf{x}),$$

we have both $\tau \notin LO$ and also $\neg\tau \notin LO$. Similarly, some linear orderings have a minimum elements, and some do not. Suppose that we start with our three linear ordering axioms $\sigma_1, \sigma_2, \sigma_3$, and then add two axioms σ_4 and σ_5 saying that there is no minimum element and there is no maximum element. The theory $Cn(\{\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5\})$ is called the theory of linear orderings without endpoints. It turns out that this theory is also not complete. Consider the two models $(\mathbb{Z}, <)$ and $(\mathbb{Q}, <)$. The rational ordering is *dense*, i.e. between any two elements we can always find another. In other words, we have

$$(\mathbb{Q}, <) \vDash \forall \mathsf{x} \forall \mathsf{y}(\mathsf{Rxy} \to \exists \mathsf{z}(\mathsf{Rxz} \wedge \mathsf{Rzy})).$$

However, we have

$$(\mathbb{Z}, <) \nvDash \forall \mathsf{x} \forall \mathsf{y}(\mathsf{Rxy} \to \exists \mathsf{z}(\mathsf{Rxz} \wedge \mathsf{Rzy}))$$

because there is no element between 0 and 1. Thus, the theory of linear orderings without endpoints is also not complete, because it neither contains $\forall \mathsf{x} \forall \mathsf{y}(\mathsf{Rxy} \to \exists \mathsf{z}(\mathsf{Rxz} \wedge \mathsf{Rzy}))$ nor its negation. If we add the density condition as an axiom, we obtain an important theory.

Definition 5.3.10. *Let $\mathcal{L} = \{\mathsf{R}\}$ where R is a binary relation symbol. Consider the following sentences*

$$\begin{aligned}
&\sigma_1 : \forall \mathsf{x} \neg \mathsf{Rxx} \\
&\sigma_2 : \forall \mathsf{x} \forall \mathsf{y} \forall \mathsf{z}((\mathsf{Rxy} \wedge \mathsf{Ryz}) \to \mathsf{Rxz}) \\
&\sigma_3 : \forall \mathsf{x} \forall \mathsf{y}(\mathsf{x} = \mathsf{y} \vee \mathsf{Rxy} \vee \mathsf{Ryx}) \\
&\sigma_4 : \forall \mathsf{x} \exists \mathsf{y} \mathsf{Rxy} \\
&\sigma_5 : \forall \mathsf{x} \exists \mathsf{y} \mathsf{Ryx} \\
&\sigma_6 : \forall \mathsf{x} \forall \mathsf{y}(\mathsf{Rxy} \to \exists \mathsf{z}(\mathsf{Rxz} \wedge \mathsf{Rzy}))
\end{aligned}$$

and let $DLO = Cn(\{\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5, \sigma_6, \})$. DLO is called the theory of dense (strict) linear orderings without endpoints.

Notice that $I(DLO, n) = 0$ for all $n \in \mathbb{N}^+$ because every finite linear ordering has a least element (see Example 5.3.5). Of course, there are countable models of DLO, such as $(\mathbb{Q}, <)$. Somewhat amazingly, any two countably infinite models of DLO are isomorphic.

Theorem 5.3.11 (Cantor)**.** *Suppose that $\mathcal{M}$ and $\mathcal{N}$ are two countably infinite models of DLO. We then have that $\mathcal{M} \cong \mathcal{N}$.*

Proof. Since $\mathcal{M}$ is countably infinite, we can list the elements of M without repetition as $m_0, m_1, m_2, \ldots$. Similarly, we an list the elements of N without repetition as $n_0, n_1, n_2, \ldots$. We now define a sequence of "partial isomorphisms" $h_k \colon M \to N$, i.e. each h_k will be a function from some finite subset of M to N that preserves the relation. More formally, we will have the following for each $k \in \mathbb{N}$:

- domain(h_k) is a finite nonempty set.
- Each h_k is injective.
- For each $\ell \in \mathbb{N}$, we have $\{m_0, m_1, \ldots, m_\ell\} \subseteq \text{domain}(h_{2\ell})$.
- For each $\ell \in \mathbb{N}$, we have $\{n_0, n_1, \ldots, n_\ell\} \subseteq \text{range}(h_{2\ell+1})$.
- $h_k \subseteq h_{k+1}$, i.e. whenever $a \in \text{domain}(h_k)$, we have $a \in \text{domain}(h_{k+1})$ and $h_{k+1}(a) = h_k(a)$.
- Each h_k is a partial isomorphism, i.e. for all $a, b \in \text{domain}(h_k)$, we have $(a, b) \in \mathsf{R}^{\mathcal{M}}$ if and only if $(h_k(a), h_k(b)) \in \mathsf{R}^{\mathcal{N}}$.

We start by letting h_0 be the partial function with domain $\{m_0\}$ where $h_0(m_0) = n_0$, and then we let $h_1 = h_0$ (since n_0 is already in range(h_0)). Suppose that $k \in \mathbb{N}^+$ and we have defined h_k. We have two cases.

- *Case 1:* Suppose that k is odd, and fix $\ell \in \mathbb{N}$ with $k = 2\ell - 1$. If $m_\ell \in \text{domain}(h_k)$, let $h_{k+1} = h_k$. Suppose then that $m_\ell \notin \text{domain}(h_k)$. Notice that $A = \text{domain}(h_k) \cup \{m_\ell\}$ is a finite nonempty subset of M, so when we restrict $\mathsf{R}^{\mathcal{M}}$ to this finite set, we obtain a finite linear ordering. Similarly, we have that $C = \text{range}(h_k)$ is a finite nonempty subset of N, so when we restrict $\mathsf{R}^{\mathcal{N}}$ to this finite set, we obtain a finite linear ordering.
  - *Subcase 1:* Suppose m_ℓ is the least element of the finite linear ordering A. Since C is a finite linear ordering, it has a least element, say c. Since $\mathcal{N}$ is a model of DLO, we can fix $d \in N$ with $(d, c) \in \mathsf{R}^{\mathcal{N}}$. We now extend h_k to h_{k+1} by letting $h_{k+1}(m_\ell) = d$.
  - *Subcase 2:* Suppose m_ℓ is the greatest element of the finite linear ordering A. Since C is a finite linear ordering, it has a greatest element, say c. Since $\mathcal{N}$ is a model of DLO, we can fix $d \in N$ with $(c, d) \in \mathsf{R}^{\mathcal{N}}$. We now extend h_k to h_{k+1} by letting $h_{k+1}(m_\ell) = d$.
  - *Subcase 3:* Otherwise, m_ℓ has an immediate predecessor a and immediate successor b in the finite linear ordering A. Since $(a, b) \in \mathsf{R}^{\mathcal{M}}$, we have $(h_k(a), h_k(b)) \in \mathsf{R}^{\mathcal{N}}$. As $\mathcal{N}$ is a model of DLO, we can fix $d \in N$ with $(h_k(a), d) \in \mathsf{R}^{\mathcal{N}}$ and $(d, h_k(b)) \in \mathsf{R}^{\mathcal{N}}$. We now extend h_k to h_{k+1} by letting $h_{k+1}(m_\ell) = d$.
- *Case 2:* Suppose that k is even, and fix $\ell \in \mathbb{N}$ with $k = 2\ell$. If $n_\ell \in \text{range}(h_k)$, let $h_{k+1} = h_k$. Suppose then that $n_\ell \notin \text{range}(h_k)$. Notice that $A = \text{domain}(h_k)$ is a finite nonempty subset of M, so when we restrict $\mathsf{R}^{\mathcal{M}}$ to this finite set, we obtain a finite linear ordering. Similarly, we have that $C = \text{range}(h_k) \cup \{n_\ell\}$ is a finite nonempty subset of N, so when we restrict $\mathsf{R}^{\mathcal{N}}$ to this finite set, we obtain a finite linear ordering.
  - *Subcase 1:* Suppose n_ℓ is the least element of the finite linear ordering C. Since A is a finite linear ordering, it has a least element, say a. Since $\mathcal{M}$ is a model of DLO, we can fix $b \in M$ with $(b, a) \in \mathsf{R}^{\mathcal{M}}$. We now extend h_k to h_{k+1} by letting $h_{k+1}(b) = n_\ell$.
  - *Subcase 2:* Suppose n_ℓ is the greatest element of the finite linear ordering C. Since A is a finite linear ordering, it has a greatest element, say a. Since $\mathcal{M}$ is a model of DLO, we can fix $b \in M$ with $(a, b) \in \mathsf{R}^{\mathcal{M}}$. We now extend h_k to h_{k+1} by letting $h_{k+1}(b) = n_\ell$.
  - *Subcase 3:* Otherwise, n_ℓ has an immediate predecessor c and immediate successor d in the finite linear ordering C. Fix $a, b \in M$ with $h(a) = c$ and $h(b) = d$. Since $(c, d) \in \mathsf{R}^{\mathcal{N}}$, we have $(h(a), h(b)) \in \mathsf{R}^{\mathcal{N}}$, so $(a, b) \in \mathsf{R}^{\mathcal{M}}$. As $\mathcal{M}$ is a model of DLO, we can fix $x \in M$ with $(a, x) \in \mathsf{R}^{\mathcal{M}}$ and $(x, b) \in \mathsf{R}^{\mathcal{M}}$. We now extend h_k to h_{k+1} by letting $h_{k+1}(x) = n_\ell$.

Now it is straightforward to check that if h_k satisfies all of the above conditions, then h_{k+1} also satisfies all of the necessary conditions, regardless of which subcase we take.

Now define $h\colon M \to N$ by letting $h(m_\ell) = h_{2\ell}(m_\ell)$ for each $\ell \in \mathbb{N}$. Using the second through fifth conditions on the h_k, we conclude that h is a bijection. Now let $a, b \in M$ be arbitrary. Fix $k, \ell \in \mathbb{N}$ with $a = m_k$ and $b = m_\ell$. Let $t = \max\{k, \ell\}$. Since $a, b \in \text{domain}(h_{2t})$, we have $(a, b) \in \mathsf{R}^{\mathcal{M}}$ if and only if $(h_{2t}(a), h_{2t}(b)) \in \mathsf{R}^{\mathcal{M}}$, which by fifth condition on the h_k is if and only if $(h(a), h(b)) \in \mathsf{R}^{\mathcal{M}}$. Therefore, h is an isomorphism. □

Proposition 5.3.12. *Let T be a theory. The following are equivalent:*

1. *T is complete.*
2. *Every two models of T are elementarily equivalent.*

Proof. Suppose that T is not complete. Fix $\sigma \in Sent_{\mathcal{L}}$ such that $\sigma \notin T$ and also $\neg\sigma \notin T$. Since T is theory, we have both $T \nvDash \sigma$ and also $T \nvDash \neg\sigma$. Since $T \nvDash \sigma$, we can fix a model $\mathcal{M}$ of $T \cup \{\neg\sigma\}$. Since $T \nvDash \neg\sigma$, we can fix a model $\mathcal{N}$ of $T \cup \{\sigma\}$. Now $\mathcal{M} \vDash \neg\sigma$ and $\mathcal{N} \vDash \sigma$, so $\mathcal{M} \not\equiv \mathcal{N}$. Thus, there exist two models of T that are not elementarily equivalent.

Suppose that T is complete, and let $\mathcal{M}$ and $\mathcal{N}$ be two arbitrary models of T. Let $\sigma \in Sent_{\mathcal{L}}$ be arbitrary. If $\sigma \in T$, we then have that both $\mathcal{M} \vDash \sigma$ and $\mathcal{N} \vDash \sigma$. Suppose that $\sigma \notin T$. Since T is complete, we then have that $\neg\sigma \in T$, hence $\mathcal{M} \vDash \neg\sigma$ and $\mathcal{N} \vDash \neg\sigma$. It follows that both $\mathcal{M} \nvDash \sigma$ and $\mathcal{N} \nvDash \sigma$. Therefore, for all $\sigma \in Sent_{\mathcal{L}}$, we have that $\mathcal{M} \vDash \sigma$ if and only if $\mathcal{N} \vDash \sigma$, so $\mathcal{M} \equiv \mathcal{N}$. □

Theorem 5.3.13 (Countable Łos-Vaught Test)**.** *Let $\mathcal{L}$ be a countable language. Suppose that T is an $\mathcal{L}$-theory such that all models of T are infinite, and suppose also that every two countably infinite models of T are isomorphic. We then have that T is complete.*

Proof. We show that any two models of T are elementarily equivalent. Let $\mathcal{M}_1$ and $\mathcal{M}_2$ be two arbitrary models of T. By the Countable Lowenheim-Skolem-Tarski Theorem, we can fix countable elementary substructures $\mathcal{N}_1 \preceq \mathcal{M}_1$ and $\mathcal{N}_2 \preceq \mathcal{M}_2$. Now $\mathcal{N}_1$ and $\mathcal{N}_2$ are also both models of T, are hence are both infinite by assumption. Since any two countably infinite models of T are isomorphic, we conclude that $\mathcal{N}_1 \cong \mathcal{N}_2$, and hence $\mathcal{N}_1 \equiv \mathcal{N}_2$ by Corollary 4.3.9. Now since each $\mathcal{N}_i$ is an elementary substructure of $\mathcal{M}_i$, we also have both $\mathcal{M}_1 \equiv \mathcal{N}_1$ and $\mathcal{M}_2 \equiv \mathcal{N}_2$. Therefore, $\mathcal{M}_1 \equiv \mathcal{M}_2$. □

Corollary 5.3.14. *DLO is complete.*

Proof. Immediate from Theorem 5.3.11 and the Countable Łos-Vaught Test, together with the fact that *DLO* has no finite models (since a finite linear ordering has a least element) □

Corollary 5.3.15. *In the language $\mathcal{L} = \{\mathsf{R}\}$ where R is a binary relation symbol, we have $(\mathbb{Q}, <) \equiv (\mathbb{R}, <)$.*

Proof. Both $(\mathbb{Q}, <)$ and $(\mathbb{R}, <)$ are models of *DLO*, so this follows from Corollary 5.3.14 and Proposition 5.3.12. □

5.4 Equivalent Formulas

Given formulas φ and ψ, we use $\varphi \leftrightarrow \psi$ as shorthand for $(\varphi \to \psi) \wedge (\psi \to \varphi)$.

Definition 5.4.1. *Let $\mathcal{L}$ be a language, and let $\varphi, \psi \in Form_{\mathcal{L}}$. We say that φ and ψ are semantically equivalent if $\varphi \vDash \psi$ and $\psi \vDash \varphi$. This is equivalent to saying that $\emptyset \vDash \varphi \leftrightarrow \psi$.*

We now list a bunch of simple rules for manipulating formulas the preserve semantic equivalence.

Proposition 5.4.2. *Let $\varphi_1, \varphi_2, \psi_1, \psi_2 \in Form_{\mathcal{L}}$, and suppose that $\emptyset \vDash \varphi_1 \leftrightarrow \psi_1$ and that $\emptyset \vDash \varphi_2 \leftrightarrow \psi_2$. We have the following:*

1. $\emptyset \vDash (\neg\varphi_1) \leftrightarrow (\neg\psi_1)$.
2. $\emptyset \vDash \exists \mathsf{x}\varphi_1 \leftrightarrow \exists \mathsf{x}\psi_1$.
3. $\emptyset \vDash \forall \mathsf{x}\varphi_1 \leftrightarrow \forall \mathsf{x}\psi_1$.
4. $\emptyset \vDash (\varphi_1 \wedge \varphi_2) \leftrightarrow (\psi_1 \wedge \psi_2)$.
5. $\emptyset \vDash (\varphi_1 \vee \varphi_2) \leftrightarrow (\psi_1 \vee \psi_2)$.
6. $\emptyset \vDash (\varphi_1 \to \varphi_2) \leftrightarrow (\psi_1 \to \psi_2)$.

Proposition 5.4.3. *For all $\varphi, \psi \in Form_{\mathcal{L}}$, we have the following:*

1. $\emptyset \vDash \neg(\exists \mathsf{x}\varphi) \leftrightarrow \forall \mathsf{x}(\neg\varphi)$.
2. $\emptyset \vDash \neg(\forall \mathsf{x}\varphi) \leftrightarrow \exists \mathsf{x}(\neg\varphi)$.
3. *If $\mathsf{x} \notin FreeVar(\psi)$, then $\emptyset \vDash (\exists \mathsf{x}\varphi) \wedge \psi \leftrightarrow \exists \mathsf{x}(\varphi \wedge \psi)$.*
4. *If $\mathsf{x} \notin FreeVar(\psi)$, then $\emptyset \vDash (\forall \mathsf{x}\varphi) \wedge \psi \leftrightarrow \forall \mathsf{x}(\varphi \wedge \psi)$.*
5. *If $\mathsf{x} \notin FreeVar(\psi)$, then $\emptyset \vDash (\exists \mathsf{x}\varphi) \vee \psi \leftrightarrow \exists \mathsf{x}(\varphi \vee \psi)$.*
6. *If $\mathsf{x} \notin FreeVar(\psi)$, then $\emptyset \vDash (\forall \mathsf{x}\varphi) \vee \psi \leftrightarrow \forall \mathsf{x}(\varphi \vee \psi)$.*
7. *If $\mathsf{x} \notin FreeVar(\psi)$, then $\emptyset \vDash (\exists \mathsf{x}\varphi) \to \psi \leftrightarrow \forall \mathsf{x}(\varphi \to \psi)$.*
8. *If $\mathsf{x} \notin FreeVar(\psi)$, then $\emptyset \vDash (\forall \mathsf{x}\varphi) \to \psi \leftrightarrow \exists \mathsf{x}(\varphi \to \psi)$.*

Proposition 5.4.4. *For any $\varphi \in Form_{\mathcal{L}}$ and $\mathsf{x} \in Var$, we have the following:*

1. *If $\mathsf{y} \notin OccurVar(\varphi)$, then $\emptyset \vDash \exists \mathsf{x}\varphi \leftrightarrow \exists \mathsf{y}(\varphi^{\mathsf{y}}_{\mathsf{x}})$.*
2. *If $\mathsf{y} \notin OccurVar(\varphi)$, then $\emptyset \vDash \forall \mathsf{x}\varphi \leftrightarrow \forall \mathsf{y}(\varphi^{\mathsf{y}}_{\mathsf{x}})$.*

Definition 5.4.5. *Let $\mathcal{L}$ be a language. A* literal *is either an atomic formula over $\mathcal{L}$, or the negation of an atomic formula. We let $Literal_{\mathcal{L}}$ be the set of literals.*

Definition 5.4.6. *Let $\mathcal{L}$ be a set. We let $Conj_P = G(Sym^*_{\mathcal{L}}, Lit_{\mathcal{L}}, \{h_\wedge\})$ be the formulas obtained by starting with the literals and generating using only $h_\wedge$, and call $Conj_{\mathcal{L}}$ the set of* conjunctive formulas. *From here, we define $DNF_{\mathcal{L}} = G(Sym^*_{\mathcal{L}}, Conj_{\mathcal{L}}, \{h_\vee\})$ to be the formulas obtained by starting with the conjunctive formulas, and generating using only $h_\vee$. The elements of $DNF_{\mathcal{L}}$ are said to be in* disjunctive normal form.

Proposition 5.4.7. *Suppose that $\varphi(\mathsf{x}_1, \mathsf{x}_2, \ldots, \mathsf{x}_k) \in Form_{\mathcal{L}}$ is quantifier-free. There exists a quantifier-free formula $\theta(\mathsf{x}_1, \mathsf{x}_2, \ldots, \mathsf{x}_k)$ in disjunctive normal form such that $\emptyset \vDash \varphi \leftrightarrow \theta$.*

Proof. As in Proposition 3.2.11, it's possible to show by induction that every quantifier-free formula is semantically equivalent to one built up by starting with literals, and then generated using $\wedge$ and $\vee$ (here we are using the fact that $\varphi \to \psi$ is semantically equivalent to $(\neg\varphi) \vee \psi$, that $\neg(\varphi \wedge \psi)$ is semantically equivalent to $(\neg\varphi) \vee (\neg\psi)$, and that $\neg(\varphi \vee \psi)$ is semantically equivalent to $(\neg\varphi) \wedge (\neg\psi)$). Now we can use the fact that $\varphi \wedge (\psi \vee \gamma)$ is semantically equivalent to $(\varphi \wedge \psi) \vee (\varphi \wedge \gamma)$ and that $(\varphi \vee \psi) \wedge \gamma$ is semantically equivalent to $(\varphi \wedge \gamma) \vee (\psi \wedge \gamma)$ to push the $\wedge$ connectives to the inside. $\square$

Definition 5.4.8. *A formula φ is called a* prenex formula *if it is an element of*

$$G(Sym^*_{\mathcal{L}}, QuantFreeForm_{\mathcal{L}}, \{h_{\forall,\mathsf{x}}, h_{\exists,\mathsf{x}} : \mathsf{x} \in Var\}).$$

In other words, a prenex formula is a quantifier-free formula with a bock of quantifiers (potentially mixed $\exists$ and $\forall$ quantifiers) at the front.

Proposition 5.4.9. *For every $\varphi \in Form_{\mathcal{L}}$, there exists a prenex formula ψ such that $\emptyset \vDash \varphi \leftrightarrow \psi$.*

Proof. Repeatedly apply Proposition 5.4.4 and Proposition 5.4.3 to move all of the quantifiers to the front of the formula. □

5.5 Quantifier Elimination

In the previous section, we showed how to transform formulas into semantically equivalent ones that had a particularly simple "structure". In that setting, the equivalence was relative to the empty set. In other words, the two formulas had to have the same meaning for *every* choice of $\mathcal{L}$-structure and variable assignment. What if we allow ourselves to include sets of formulas on the left to help us simplify the formulas even more?

For example, consider putting a theory T on the left of $\vDash$. Now a theory is a set of sentences, but we can still consider putting formulas with free variable on the right-hand side of $\vDash$. In such circumstances, we have to think about variable assignments as well, but the hope is that we can find a "simpler" equivalent formula to a given one. For example, let $\mathcal{L} = \{\mathsf{R}\}$, where R is a binary relation symbol. Notice that

$$\emptyset \nvDash \exists \mathsf{x}(\mathsf{Rax} \wedge \mathsf{Rbx}) \leftrightarrow \mathsf{Rab}$$

because we can let $\mathcal{M}$ be the $\mathcal{L}$-structure with $M = \{0, 1, 2\}$ and $\mathsf{R}^{\mathcal{M}} = \{(0, 2), (1, 2)\}$, and then

$$(\mathcal{M}, 0, 1) \vDash \exists \mathsf{x}(\mathsf{Rax} \wedge \mathsf{Rbx})$$

but

$$(\mathcal{M}, 0, 1) \nvDash \mathsf{Rab},$$

so

$$(\mathcal{M}, 0, 1) \nvDash \exists \mathsf{x}(\mathsf{Rax} \wedge \mathsf{Rbx}) \leftrightarrow \mathsf{Rab}).$$

Even though these formulas are not equivalent over the theory $Cn(\emptyset)$, it turns out that they are equivalent over the theory of equivalence relations. Let $EqRel$ is the theory of equivalence relations, i.e. $EqRel = Cn(\sigma_1, \sigma_2, \sigma_3)$ where the σ_i express that the relation is reflexive, symmetric, and transitive. We then have that

$$EqRel \vDash \exists \mathsf{x}(\mathsf{Rax} \wedge \mathsf{Rbx}) \leftrightarrow \mathsf{Rab}.$$

In other words, in every model $(\mathcal{M}, s)$ of $EqRel$, we have that

$$(\mathcal{M}, s) \vDash \exists \mathsf{x}(\mathsf{Rax} \wedge \mathsf{Rbx}) \leftrightarrow \mathsf{Rab}.$$

Thus, relative to the theory $EqRel$, the formula $\exists \mathsf{x}(\mathsf{Rax} \wedge \mathsf{Rbx})$, which has a quantifier and two free variables, is equivalent to the quantifier-free formula Rab in the same free variables. Notice that if we work with DLO instead of $EqRel$, then we have

$$DLO \vDash \exists \mathsf{x}(\mathsf{Rax} \wedge \mathsf{Rbx}) \leftrightarrow (\mathsf{a} = \mathsf{a}) \wedge (\mathsf{b} = \mathsf{b}).$$

For another example, consider solving linear equations. If we are working in the real numbers, then we can always solve the equation $ax + b = 0$, unless $a = 0$ and $b \neq 0$. More formally, let $\mathcal{L} = \{0, 1, +, -, \cdot\}$ be

the language of ring theory. If we let $\mathcal{M}$ be the ring $\mathbb{R}$, and let $\mathsf{a}, \mathsf{b} \in Var$, then for any variable assignment $s\colon Var \to \mathbb{R}$, we have

$$(\mathcal{M}, s) \vDash \exists \mathsf{x}(\mathsf{a} \cdot \mathsf{x} + \mathsf{b} = \mathsf{0}) \leftrightarrow (\neg(\mathsf{a} = \mathsf{0}) \vee \mathsf{b} = \mathsf{0}).$$

Notice that

$$\emptyset \nvDash \exists \mathsf{x}(\mathsf{a} \cdot \mathsf{x} + \mathsf{b} = \mathsf{0}) \leftrightarrow (\neg(\mathsf{a} = \mathsf{0}) \vee \mathsf{b} = \mathsf{0}).$$

In fact, if R is the theory of rings, then we still have

$$R \nvDash \exists \mathsf{x}(\mathsf{a} \cdot \mathsf{x} + \mathsf{b} = \mathsf{0}) \leftrightarrow (\neg(\mathsf{a} = \mathsf{0}) \vee \mathsf{b} = \mathsf{0}),$$

because we can let $\mathcal{M}$ be the ring $\mathbb{Z}$, and notice that we have

$$(\mathcal{M}, 2, 1) \nvDash \exists \mathsf{x}(\mathsf{a} \cdot \mathsf{x} + \mathsf{b} = \mathsf{0}) \leftrightarrow (\neg(\mathsf{a} = \mathsf{0}) \vee \mathsf{b} = \mathsf{0}).$$

However, if F is the theory of fields, then we do have

$$F \vDash \exists \mathsf{x}(\mathsf{a} \cdot \mathsf{x} + \mathsf{b} = \mathsf{0}) \leftrightarrow (\neg(\mathsf{a} = \mathsf{0}) \vee \mathsf{b} = \mathsf{0}).$$

Again, relative to a sufficiently strong theory, we can find a quantifier-free equivalent to a formula with two free variables.

Definition 5.5.1. *Let T be a theory. We say that T has* quantifier elimination*, or has QE, if for every $k \geq 1$ and every $\varphi(\mathsf{x}_1, \mathsf{x}_2, \ldots, \mathsf{x}_k) \in Form_{\mathcal{L}}$, there exists a quantifier-free $\psi(\mathsf{x}_1, \mathsf{x}_2, \ldots, \mathsf{x}_k)$ such that*

$$T \vDash \varphi \leftrightarrow \psi.$$

This seems like an awful lot to ask of a theory. However, it is a pleasant surprise that several natural and important theories have QE, and in several more cases we can obtain a theory with QE by only adding a few things to the language. We first prove that it suffices to eliminate one quantifier from formulas of a very specific form.

Proposition 5.5.2. *Let T be a theory. The following are equivalent*

1. *T has QE.*
2. *For each formula $\varphi(\mathsf{x}_1, \mathsf{x}_2, \ldots, \mathsf{x}_k, \mathsf{y})$ that is a conjunction of literals, each of which has y as a free variable, there exists a quantifier-free $\psi(\mathsf{x}_1, \mathsf{x}_2, \ldots, \mathsf{x}_k)$ such that*

$$T \vDash (\exists \mathsf{y} \varphi) \leftrightarrow \psi.$$

Proof. The idea is to put a general formula in prenex form. If the innermost quantifier is $\forall$, change it to $\neg\exists\neg$ so that the innermost block is a existential quantifier followed by a quantifier-free formula φ. Now find a formula θ that is in disjunctive normal form that is semantically equivalent to φ. From here, the key fact to use is that

$$\emptyset \vDash \exists \mathsf{x}(\theta_1 \vee \theta_2) \leftrightarrow (\exists \mathsf{x} \theta_1) \vee (\exists \mathsf{x} \theta_2).$$

We can then use the assumption to find quantifier-free equivalents (relative to T) of each formulas that is an existential quantifier followed by a conjunction of literals. Now that we have eliminated the innermost quantifier, we can continue in turn to eliminate later quantifiers. □

We now do the hard work of proving QE for two specific theories. We start with the very basic theory of infinite structures in the empty language.

Theorem 5.5.3. *Let $\mathcal{L} = \emptyset$. For each $n \in \mathbb{N}^+$, let σ_n be the sentence*

$$\exists \mathsf{x}_1 \exists \mathsf{x}_2 \cdots \exists \mathsf{x}_n \bigwedge_{1 \leq i < j \leq n} \neg(\mathsf{x}_i = \mathsf{x}_j)$$

Let $T = Cn(\{\sigma_n : n \in \mathbb{N}^+\})$. T has QE.

Proof. Suppose that $k \geq 1$ and we have have a formula $\varphi(\mathsf{x}_1, \mathsf{x}_2, \ldots, \mathsf{x}_k, \mathsf{y})$ that is a conjunction of literals α_i, each of which has y as a free variable. We want to find a quantifier-free formula $\psi(\mathsf{x}_1, \mathsf{x}_2, \ldots, \mathsf{x}_k)$ such that

$$T \vDash \exists \mathsf{y} \varphi \leftrightarrow \psi.$$

If one of the literals is $\mathsf{y} = \mathsf{x}_j$ (or $\mathsf{x}_j = \mathsf{y}$) for some j, then

$$T \vDash \exists \mathsf{y} \varphi \leftrightarrow \varphi_{\mathsf{y}}^{\mathsf{x}_j}.$$

Suppose then that none of the literals is of the form $\mathsf{y} = \mathsf{x}_j$. We can remove any literals of the form $\mathsf{y} = \mathsf{y}$, and if there is a literal of the form $\neg(\mathsf{y} = \mathsf{y})$, then the formula is trivially equivalent to $\neg(\mathsf{x}_1 = \mathsf{x}_1)$. Thus, we need only examine the case where every literal is of the form $\neg(\mathsf{y} = \mathsf{x}_j)$ or $\neg(\mathsf{x}_j = \mathsf{y})$ for some j. We then have

$$T \vDash \exists \mathsf{y} \varphi \leftrightarrow (\mathsf{x}_1 = \mathsf{x}_1) \wedge (\mathsf{x}_2 = \mathsf{x}_2) \wedge \cdots \wedge (\mathsf{x}_k = \mathsf{x}_k).$$

because every model of T has infinitely many elements. □

We next show that DLO has QE. Before diving into the general proof, we first give a specific example. Suppose that we want to find a quantifier-free equivalent to

$$\exists \mathsf{y}(\mathsf{R}\mathsf{x}_2\mathsf{y} \wedge \neg(\mathsf{y} = \mathsf{x}_3) \wedge \mathsf{R}\mathsf{y}\mathsf{x}_4 \wedge \neg(\mathsf{R}\mathsf{x}_1\mathsf{y})).$$

We begin by noticing that $DLO \vDash \neg(\mathsf{R}\mathsf{x}_1\mathsf{y}) \leftrightarrow ((\mathsf{y} = \mathsf{x}_1) \vee \mathsf{R}\mathsf{y}\mathsf{x}_1)$, so we want to find a quantifier-free equivalent to

$$\exists \mathsf{y}(\mathsf{R}\mathsf{x}_2\mathsf{y} \wedge \neg(\mathsf{y} = \mathsf{x}_3) \wedge \mathsf{R}\mathsf{y}\mathsf{x}_4 \wedge ((\mathsf{y} = \mathsf{x}_1) \vee \mathsf{R}\mathsf{y}\mathsf{x}_1)).$$

Since $\wedge$ distributes over $\vee$, it suffices to find a quantifier-free equivalent to

$$\exists \mathsf{y}((\mathsf{R}\mathsf{x}_2\mathsf{y} \wedge \neg(\mathsf{y} = \mathsf{x}_3) \wedge \mathsf{R}\mathsf{y}\mathsf{x}_4 \wedge (\mathsf{y} = \mathsf{x}_1)) \vee ((\mathsf{R}\mathsf{x}_2\mathsf{y} \wedge \neg(\mathsf{y} = \mathsf{x}_3) \wedge \mathsf{R}\mathsf{y}\mathsf{x}_4 \wedge \mathsf{R}\mathsf{y}\mathsf{x}_1))).$$

Since this last formula is equivalent to

$$\exists \mathsf{y}(\mathsf{R}\mathsf{x}_2\mathsf{y} \wedge \neg(\mathsf{y} = \mathsf{x}_3) \wedge \mathsf{R}\mathsf{y}\mathsf{x}_4 \wedge (\mathsf{y} = \mathsf{x}_1)) \vee \exists \mathsf{y}(\mathsf{R}\mathsf{x}_2\mathsf{y} \wedge \neg(\mathsf{y} = \mathsf{x}_3) \wedge \mathsf{R}\mathsf{y}\mathsf{x}_4 \wedge \mathsf{R}\mathsf{y}\mathsf{x}_1),$$

it suffices to find a quantifier-free equivalent to each of

$$\exists \mathsf{y}(\mathsf{R}\mathsf{x}_2\mathsf{y} \wedge \neg(\mathsf{y} = \mathsf{x}_3) \wedge \mathsf{R}\mathsf{y}\mathsf{x}_4 \wedge (\mathsf{y} = \mathsf{x}_1))$$

and

$$\exists \mathsf{y}(\mathsf{R}\mathsf{x}_2\mathsf{y} \wedge \neg(\mathsf{y} = \mathsf{x}_3) \wedge \mathsf{R}\mathsf{y}\mathsf{x}_4 \wedge \mathsf{R}\mathsf{y}\mathsf{x}_1).$$

Now we have

$$DLO \vDash \exists \mathsf{y}(\mathsf{R}\mathsf{x}_2\mathsf{y} \wedge \neg(\mathsf{y} = \mathsf{x}_3) \wedge \mathsf{R}\mathsf{y}\mathsf{x}_4 \wedge (\mathsf{y} = \mathsf{x}_1)) \leftrightarrow (\mathsf{R}\mathsf{x}_2\mathsf{x}_1 \wedge \neg(\mathsf{x}_1 = \mathsf{x}_3) \wedge \mathsf{R}\mathsf{x}_1\mathsf{x}_4),$$

and we have

$$DLO \vDash \exists \mathsf{y}(\mathsf{R}\mathsf{x}_2\mathsf{y} \wedge \neg(\mathsf{y} = \mathsf{x}_3) \wedge \mathsf{R}\mathsf{y}\mathsf{x}_4 \wedge \mathsf{R}\mathsf{y}\mathsf{x}_1) \leftrightarrow (\mathsf{R}\mathsf{x}_2\mathsf{x}_4 \wedge \mathsf{R}\mathsf{x}_2\mathsf{x}_1),$$

where we use the fact that if $a < b$ in a model of DLO, then there are infinitely many c with $a < c < b$. Thus, our original formula is equivalent over DLO to

$$(\mathsf{R}\mathsf{x}_2\mathsf{x}_1 \wedge \neg(\mathsf{x}_1 = \mathsf{x}_3) \wedge \mathsf{R}\mathsf{x}_1\mathsf{x}_4) \vee (\mathsf{R}\mathsf{x}_2\mathsf{x}_4 \wedge \mathsf{R}\mathsf{x}_2\mathsf{x}_1).$$

We generalize this example in the following proof.

Theorem 5.5.4. *DLO has QE.*

Proof. Suppose that we have have a formula $\varphi(\mathsf{x}_1, \mathsf{x}_2, \ldots, \mathsf{x}_k, \mathsf{y})$ that is a conjunction of literals α_i, each of which has y as a free variable. We want to find a quantifier-free formula $\psi(\mathsf{x}_1, \mathsf{x}_2, \ldots, \mathsf{x}_k)$ such that

$$DLO \vDash \exists \mathsf{y} \varphi \leftrightarrow \psi.$$

If one of the literals is $\mathsf{y} = \mathsf{x}_j$ (or $\mathsf{x}_j = \mathsf{y}$) for some j, then

$$DLO \vDash \exists \mathsf{y} \varphi \leftrightarrow \varphi_{\mathsf{y}}^{\mathsf{x}_j}.$$

Suppose then that none of the literals is of the form $\mathsf{y} = \mathsf{x}_j$. If any of the literals is of the form $\neg \mathsf{R}\mathsf{x}_j\mathsf{y}$, we can replace it with $(\mathsf{y} = \mathsf{x}_j) \vee \mathsf{R}\mathsf{y}\mathsf{x}_j$, distribute the various $\wedge$ over the $\vee$, distribute the $\exists$ over $\vee$, and find quantifier-free equivalents to the two resulting clauses separately (as in the previous example). Similarly, if any of the literals is of the form $\neg \mathsf{R}\mathsf{y}\mathsf{x}_j$, we can replace it with $(\mathsf{y} = \mathsf{x}_j) \vee \mathsf{R}\mathsf{x}_j\mathsf{y}$. Thus, we may assume that all of the literals are of the form $\neg(\mathsf{y} = \mathsf{x}_j)$, $\mathsf{R}\mathsf{y}\mathsf{x}_j$, or $\mathsf{R}\mathsf{x}_j\mathsf{y}$. Let

- $L = \{j \in \{1, 2, \ldots, k\} : \text{There exists an } \alpha_i \text{ equal to } \mathsf{R}\mathsf{x}_j\mathsf{y}\}$.
- $U = \{j \in \{1, 2, \ldots, k\} : \text{There exists an } \alpha_i \text{ equal to } \mathsf{R}\mathsf{y}\mathsf{x}_j\}$.

Now if either L or U is empty, then

$$DLO \vDash \exists \mathsf{y} \varphi \leftrightarrow \mathsf{x}_1 = \mathsf{x}_1$$

because, if $U = \emptyset$ say, we need only notice that in any model $\mathcal{M}$ of DLO together with $c_1, c_2, \ldots, c_k \in M$, there are infinitely many $d \in M$ such that $(c_i, d) \in \mathsf{R}^{\mathcal{M}}$ for all i.

Suppose then that both $L \neq \emptyset$ and $U \neq \emptyset$. We claim that

$$DLO \vDash \exists \mathsf{y} \varphi \leftrightarrow \bigwedge_{\ell \in L} \bigwedge_{u \in U} \mathsf{R}\mathsf{x}_\ell\mathsf{x}_u$$

To see this, consider an arbitrary model $\mathcal{M}$ of DLO together with $c_1, c_2, \ldots, c_k \in M$.

- Assume that there exists a $d \in M$ with $(c_\ell, d) \in \mathsf{R}^{\mathcal{M}}$ for all $\ell \in L$ and $(d, c_u) \in \mathsf{R}^{\mathcal{M}}$ for all $u \in U$. We then have $(c_\ell, c_u) \in \mathsf{R}^{\mathcal{M}}$ for all $\ell \in L$ and $u \in U$ by transitivity.
- For the converse, assume that we know that $(c_\ell, c_u) \in \mathsf{R}^{\mathcal{M}}$ for all $\ell \in L$ and $u \in U$. Since $\mathcal{M}$ is a linear ordering, there exists $\ell^* \in L$ with (c_ℓ, c_{ℓ^*}) for all $\ell \in L$. Similarly, there exists $u^* \in U$ with (c_{u^*}, c_u) for all $u \in U$. By assumption, we then have that $(c_{\ell^*}, c_{u^*}) \in \mathsf{R}^{\mathcal{M}}$. Now the DLO axioms imply that there exists infinitely many $a \in M$ with $(c_{\ell^*}, a) \in \mathsf{R}^{\mathcal{M}}$ and $(a, c_{u^*}) \in \mathsf{R}^{\mathcal{M}}$. Thus, we can fix such an a with $a \neq c_i$ for all i, and this a will make be an existential witness for φ.

Therefore, DLO has QE. □

Notice that in our definition of QE, we assumed that $k \geq 1$. In other words, we did not require that we could always find a quantifier-free equivalent sentence for each $\sigma \in Sent_{\mathcal{L}}$. We chose to do this because if our language does not have any constant symbols (such as the language for DLO), then there simply are no quantifier-free sentences! If our language does have a constant symbol, then the proof that a theory has QE typically also applies in the case when there are no free variables. And in the case when our language does not have any constant symbols, we can perform an ugly hack by taking a sentence σ, and finding a quantifier-free equivalent to the formula $\varphi(x)$ equal to $\sigma \wedge (\mathsf{x} = \mathsf{x})$. Of course, in this case, the value of x does not affect the truth in any structure, so the truth value of the formula output by a quantifier elimination procedure must also not depend on x in any fixed structure.

What do we gain by knowing that a formula has QE? The first advantage is that it is much easier to understand the definable sets in any model.

Proposition 5.5.5. *Suppose that T is a theory with QE. Given any model $\mathcal{M}$ of T, a set $X \subseteq M^k$ is definable in $\mathcal{M}$ if and only if it is definable by a quantifier-free formula.*

Proof. Immediate. □

Corollary 5.5.6. *Let T be a theory that has QE, let $\mathcal{M}$ be a model of T, and let $k \in \mathbb{N}^+$. Let $\mathcal{Z}$ be the set of all subsets of M^k which are definable by atomic formulas. The set of definable subsets of M^k equals $G(\mathcal{P}(M^k), \mathcal{Z}, \{h_1, h_2\})$ where $h_1 \colon \mathcal{P}(M^k) \to \mathcal{P}(M^k)$ is the complement function and $h_2 \colon \mathcal{P}(M^k)^2 \to \mathcal{P}(M^k)$ is the union function.*

Proof. A quantifier-free formula is built up from atomic formulas using $\neg$, $\wedge$, $\vee$, and $\to$. Moreover, we know that every quantifier-free formula is semantically equivalent to one that only uses $\neg$ and $\vee$. Since these operations correspond to complement and union on the corresponding definable sets, we obtain the result. □

For example, in $(\mathbb{Q}, <)$, which is a model of DLO, the only atomic formulas with one free variable are $\mathsf{x} = \mathsf{x}$, $\neg(\mathsf{x} = \mathsf{x})$, $\mathsf{x} < \mathsf{x}$, and $\neg(\mathsf{x} < \mathsf{x})$. Each of these defines either $\emptyset$ or $\mathbb{Q}$. Since the collection of sets $\{\emptyset, \mathbb{Q}\}$ is closed under complement and union, we now have another proof (without using automorphisms) that $\emptyset$ and $\mathbb{Q}$ are the only definable subsets of $\mathbb{Q}$ in $(\mathbb{Q}, <)$. We can also use this method to determine the definable sets of $\mathbb{Q}^2$ in the structure $(\mathbb{Q}, <)$ (see the homework).

Another interesting consequence of a theory having QE is the following surprising fact.

Proposition 5.5.7. *Let T be a theory that has QE. Suppose that $\mathcal{A}$ and $\mathcal{M}$ are models of T and that $\mathcal{A}$ is a substructure of $\mathcal{M}$. We then have that $\mathcal{A} \preceq \mathcal{M}$.*

Proof. Let $\varphi \in Form_{\mathcal{L}}$ and let $s \colon Var \to A$ be a variable assignment. Suppose first that $\varphi \notin Sent_{\mathcal{L}}$. Since T has QE, we may fix a quantifier-free $\psi \in Form_{\mathcal{L}}$ such that $T \vDash \varphi \leftrightarrow \psi$. We then have

$$\begin{aligned} (\mathcal{M}, s) \vDash \varphi &\Leftrightarrow (\mathcal{M}, s) \vDash \psi && \text{(since } \mathcal{M} \text{ is a model of } T\text{)} \\ &\Leftrightarrow (\mathcal{A}, s) \vDash \psi && \text{(by Corollary 4.3.12)} \\ &\Leftrightarrow (\mathcal{A}, s) \vDash \varphi && \text{(since } \mathcal{A} \text{ is a model of } T\text{).} \end{aligned}$$

If $\sigma \in Sent_{\mathcal{L}}$, then we can use the hack alluded to above. That is, let $\varphi(\mathsf{x})$ be the formula $\sigma \wedge (\mathsf{x} = \mathsf{x})$. Since T has QE, we may fix a quantifier-free $\psi(\mathsf{x}) \in Form_{\mathcal{L}}$ such that $T \vDash \varphi \leftrightarrow \psi$. Now fix some $a \in A$. By the above argument, we then have that $(\mathcal{M}, s) \vDash \varphi$ if and only if $(\mathcal{A}, s) \vDash \varphi$. Now notice that we trivially have $(\mathcal{M}, s) \vDash \sigma$ if and only if $(\mathcal{M}, s) \vDash \varphi$, and similarly that $(\mathcal{A}, s) \vDash \sigma$ if and only if $(\mathcal{A}, s) \vDash \varphi$. Therefore, we conclude that $(\mathcal{M}, s) \vDash \sigma$ if and only if $(\mathcal{A}, s) \vDash \sigma$. □

In particular, since DLO has QE, we now know that $(\mathbb{Q}, <) \preceq (\mathbb{R}, <)$. Recall that we already established that $(\mathbb{Q}, <) \equiv (\mathbb{R}, <)$, but this new result is stronger.

By a similar argument, we can also use QE in an interesting way to find connections between two models that share a common substructure. We will prove the converse of this result in Theorem 7.4.1.

Proposition 5.5.8. *Let T be a theory that has QE. Suppose that $\mathcal{M}$ and $\mathcal{N}$ are models of T, and that $\mathcal{M}$ and $\mathcal{N}$ have a common substructure $\mathcal{A}$ (note that we are not assuming that $\mathcal{A}$ is a model of T). For all $\varphi \in Form_{\mathcal{L}}$ and $s \colon Var \to A$, we have $(\mathcal{M}, s) \vDash \varphi$ if and only if $(\mathcal{N}, s) \vDash \varphi$.*

Proof. Let $\varphi \in Form_{\mathcal{L}}$ be arbitrary and $s \colon Var \to A$ be arbitrary. First, assume that φ is not a sentence. Since T has QE, we can fix a quantifier-free ψ with $T \vDash \varphi \leftrightarrow \psi$. We then have

$$\begin{aligned} (\mathcal{M}, s) \vDash \varphi &\Leftrightarrow (\mathcal{M}, s) \vDash \psi && \text{(since } \mathcal{M} \text{ is a model of } T\text{)} \\ &\Leftrightarrow (\mathcal{A}, s) \vDash \psi && \text{(by Corollary 4.3.12)} \\ &\Leftrightarrow (\mathcal{N}, s) \vDash \psi && \text{(by Corollary 4.3.12)} \\ &\Leftrightarrow (\mathcal{N}, s) \vDash \varphi && \text{(since } \mathcal{N} \text{ is a model of } T\text{)} \end{aligned}$$

If φ is a sentence, then we can argue as in the previous result. □

The final application of using QE using the same ideas is to show that certain theories are complete. QE itself is not sufficient, but a very mild additional assumption gives us what we want.

Proposition 5.5.9. *Let T be a theory that has QE. If there exists an $\mathcal{L}$-structure $\mathcal{N}$ such that for every model $\mathcal{M}$ of T there is an embedding $h\colon N \to M$ from $\mathcal{N}$ to $\mathcal{M}$, then T is complete. (Notice, there is no assumption that $\mathcal{N}$ is a model of T.)*

Proof. Fix an $\mathcal{L}$-structure $\mathcal{N}$ such that for every model $\mathcal{M}$ of T there is an embedding $h\colon N \to M$ from $\mathcal{N}$ to $\mathcal{M}$, and fix $n \in N$. Let $\mathcal{M}_1$ and $\mathcal{M}_2$ be two models of T. Fix embeddings $h_1\colon N \to M_1$ and $h_2\colon N \to M_2$. For each i, let $A_i = \text{range}(h_i)$, and notice that A_i is the universe of a substructure $\mathcal{A}_i$ of $\mathcal{M}_i$. Furthermore, notice that h_i is an isomorphism from $\mathcal{N}$ to $\mathcal{A}_i$.

Let $\sigma \in Sent_{\mathcal{L}}$ and let $\varphi(\mathsf{x}) \in Form_{\mathcal{L}}$ be the formula $\sigma \wedge (\mathsf{x} = \mathsf{x})$. Since T has QE, we may fix a quantifier-free $\psi(\mathsf{x}) \in Form_{\mathcal{L}}$ such that $T \vDash \varphi \leftrightarrow \psi$. We then have

$$\begin{aligned}
\mathcal{M}_1 \vDash \sigma &\Leftrightarrow (\mathcal{M}_1, h_1(n)) \vDash \varphi \\
&\Leftrightarrow (\mathcal{M}_1, h_1(n)) \vDash \psi && \text{(since } \mathcal{M}_1 \text{ is a model of } T\text{)} \\
&\Leftrightarrow (\mathcal{A}_1, h_1(n)) \vDash \psi && \text{(by Corollary 4.3.12)} \\
&\Leftrightarrow (\mathcal{N}, n) \vDash \psi && \text{(by Theorem 4.3.6)} \\
&\Leftrightarrow (\mathcal{A}_2, h_2(n)) \vDash \psi && \text{(by Theorem 4.3.6)} \\
&\Leftrightarrow (\mathcal{M}_2, h_2(n)) \vDash \psi && \text{(by Corollary 4.3.12)} \\
&\Leftrightarrow (\mathcal{M}_2, h_2(n)) \vDash \varphi && \text{(since } \mathcal{M}_2 \text{ is a model of } T\text{)} \\
&\Leftrightarrow \mathcal{M}_2 \vDash \sigma.
\end{aligned}$$

□

5.6 Algebraically Closed Fields

A field F is algebraically closed if every nonzero polynomial in $F[x]$ has a root in F. We can write down infinitely many first-order axioms, each one saying all polynomials of a given degree have a root. We choose to work in the language $\mathcal{L} = \{0, 1, +, -, \cdot\}$ in the language of rings.

Definition 5.6.1. *Let $\mathcal{L} = \{0, 1, +, -, \cdot\}$ be the language of rings. Let $\Sigma \subseteq Sent_{\mathcal{L}}$ be the field axioms together with the sentences*

$$\forall \mathsf{a}_0 \forall \mathsf{a}_1 \cdots \forall \mathsf{a}_n (\mathsf{a}_n \neq 0 \to \exists \mathsf{x} (\mathsf{a}_n \mathsf{x}^n + \cdots + \mathsf{a}_1 \mathsf{x} + \mathsf{a}_0 = 0))$$

for each $n \in \mathbb{N}^+$. Let $ACF = Cn(\Sigma)$. ACF_p is the theory obtained by also adding $1 + 1 + \cdots + 1 = 0$ (where there are p many 1's) to Σ, and ACF_0 is the theory obtain by adding all of $\neg(1 = 0)$, $\neg(1 + 1 = 0)$, ... to Σ.

We collect a few facts about algebraically closed fields.

- Every algebraically closed field is infinite.
- The Fundamental Theorem of Algebra is the statement that $\mathbb{C}$ is an algebraically closed field.
- The set $\overline{\mathbb{Q}}$, consisting of those elements of $\mathbb{C}$ that are algebraic over $\mathbb{Q}$ (i.e. are roots of some nonzero polynomial over $\mathbb{Q}$), is also an algebraically closed field.
- Every field can be embedded in an algebraically closed field.

Theorem 5.6.2. *ACF has QE.*

Proof Sketch. The first thing to notice is that every term corresponds to a polynomial in several variables. More formally, for all terms t with variables $\mathsf{x}_1, \mathsf{x}_2, \ldots, \mathsf{x}_k$, there exists a term u that corresponds to a polynomial in $\mathbb{Z}[\mathsf{x}_1, \mathsf{x}_2, \ldots, \mathsf{x}_n]$ such that $ACF \vDash t = u$ (in fact, t and u are equivalent over the theory of commutative rings). From here, the fundamental observation is that we can think of atomic formulas with free variables in $\{\mathsf{y}, \mathsf{x}_1, \mathsf{x}_2, \ldots, \mathsf{x}_k\}$ as equations $p(\vec{\mathsf{x}}, \mathsf{y}) = 0$ where $p(\vec{\mathsf{x}}, \mathsf{y}) \in \mathbb{Z}[\vec{\mathsf{x}}, \mathsf{y}]$ is a polynomial.

Thus, we have to find quantifier-free equivalents to formulas of the form

$$\exists \mathsf{y}[\bigwedge_{i=1}^{m}(p_i(\vec{\mathsf{x}}, \mathsf{y}) = 0) \wedge \bigwedge_{j=1}^{n}(q_j(\vec{\mathsf{x}}, \mathsf{y}) \neq 0)]$$

If we just have a bunch of negated atomic formulas, i.e. if $m = 0$, then the formula is equivalent to saying that each polynomial has a nonzero coefficient (since algebraically closed fields are infinite, and a nonzero polynomial has only finitely many roots). Thus, we can assume that $m \geq 1$. Also, if $n \geq 1$, then by letting $q(\vec{\mathsf{x}}, \mathsf{y}) = \prod_{j=1}^{n} q_j(\vec{\mathsf{x}}, \mathsf{y})$, our formula is equivalent over ACF to

$$\exists \mathsf{y}[\bigwedge_{i=1}^{m}(p_i(\vec{\mathsf{x}}, \mathsf{y}) = 0) \wedge q(\vec{\mathsf{x}}, \mathsf{y}) \neq 0].$$

Thus, we can assume that $m \geq 1$ and that $n \in \{0, 1\}$.

Suppose now that R is an integral domain, that $m \geq 2$ and that $p_1, p_2, \ldots, p_m, q \in R[y]$ listed in decreasing order of degrees. Let the leading term of p_1 be ay^n and let the leading term of p_m be by^k (in our case, R will be the polynomial ring $\mathbb{Z}[\mathsf{x}_1, \mathsf{x}_2, \ldots, \mathsf{x}_n]$). We then have that there is a simultaneous root of polynomials $p_1, p_2, \ldots, p_m$ which is not a root of q if and only if either of the following happens:

1. $b = 0$ and there is simultaneous root of the polynomials $p_1, p_2, \ldots, p_{m-1}, p_m^*$ which is not a root of q, where p_m^* is the polynomial that results by deleting the leading term of p_m.
2. $b \neq 0$ and there is a simultaneous root of the polynomials $bp_1 - ay^{n-k}p_m, p_2, \ldots, p_m$ which is not a root of q.

If there is no q, i.e. if $n = 0$, then there is a simultaneous root of polynomials $p_1, p_2, \ldots, p_m$ if and only if either of the following happens:

1. $b = 0$ and there is simultaneous root of the polynomials $p_1, p_2, \ldots, p_{m-1}, p_m^*$, where p_m^* is the polynomial that results by deleting the leading term of p_m.
2. $b \neq 0$ and there is a simultaneous root of the polynomials $bp_1 - ay^{n-k}p_m, p_2, \ldots, p_m$.

For example, we have that

$$\exists \mathsf{y}(((\mathsf{x}_1^3 + 2\mathsf{x}_1\mathsf{x}_2)\mathsf{y}^2 + (5\mathsf{x}_2 + \mathsf{x}_2^2\mathsf{x}_3)\mathsf{y} + \mathsf{x}_2 = 0) \wedge (3\mathsf{x}_2 + \mathsf{x}_1\mathsf{x}_2\mathsf{x}_3)\mathsf{y} + (\mathsf{x}_1 - \mathsf{x}_2) = 0)$$

is equivalent to the disjunction of

$$\mathsf{x}_1^3 + 2\mathsf{x}_1\mathsf{x}_2 = 0 \wedge \exists \mathsf{y}(((5\mathsf{x}_2 + \mathsf{x}_2^2\mathsf{x}_3)\mathsf{y} + \mathsf{x}_2 = 0) \wedge (3\mathsf{x}_2 + \mathsf{x}_1\mathsf{x}_2\mathsf{x}_3)\mathsf{y} + (\mathsf{x}_1 - \mathsf{x}_2) = 0)$$

and

$$\exists \mathsf{y}(((3\mathsf{x}_2 + \mathsf{x}_1\mathsf{x}_2\mathsf{x}_3)(5\mathsf{x}_2 + \mathsf{x}_2^2\mathsf{x}_3) - (\mathsf{x}_1 - \mathsf{x}_2))\mathsf{y} + (3\mathsf{x}_2 + \mathsf{x}_1\mathsf{x}_2\mathsf{x}_3)\mathsf{x}_2 = 0)$$

Repeating this, we may assume that we have a formula of the form

$$\exists \mathsf{y}[p(\vec{\mathsf{x}}, \mathsf{y}) = 0 \wedge q(\vec{\mathsf{x}}, \mathsf{y}) \neq 0]$$

or

$$\exists \mathsf{y}[p(\vec{\mathsf{x}}, \mathsf{y}) = \mathsf{0}].$$

In the latter case, then we may use the fact that in an algebraically closed field, the polynomial $a_n y^n + \cdots + a_1 y + a_0$ has a root if and only if some $a_i \neq 0$ for $i > 0$, or $a_0 = 0$. Suppose then that we are in the former case. The key fact is to use here is that if p and q are polynomials over an algebraically closed field and the degree of p is at most n, then every root of p is a root of q if and only if $p \mid q^n$.

Thus, suppose that we have two polynomials p and q, and we want to find a quantifier-free formula equivalent to $p \mid q$. Suppose that $p(y) = \sum_{i=0}^{m} a_i \mathsf{y}^i$ and that $q(y) = \sum_{j=0}^{n} b_j \mathsf{y}^j$ (where the a_i and b_j are really polynomials in $\mathsf{x}_1, \mathsf{x}_2, \ldots, \mathsf{x}_k$). Now if $m = 0$, then we have $p \mid q$ if and only if either of the following is true:

- $a_0 \neq 0$.
- $a_0 = 0$ and each $b_j = 0$.

If $n = 0$, then we have $p \mid q$ if and only if either of the following is true:

- $b_0 = 0$.
- $b_0 \neq 0$, $a_0 \neq 0$, and $a_i = 0$ for $1 \leq i \leq m$.

Suppose that $1 \leq n < m$. We then have that $p \mid q$ if and only if either of the following is true:

- Each $b_j = 0$.
- $a_i = 0$ for all i with $n < i \leq m$, and $p^* \mid q$, where p^* is the result of deleting all terms from p with degree greater than n.

Finally, suppose that $1 \leq m \leq n$. We then have that $p \mid q$ if and only if either of the following is true:

- $a_m = 0$ and $p^* \mid q$, where p^*, where p^* is the polynomial that results by deleting the leading term of p.
- $a_m \neq 0$ and $p \mid (a_m q - b_n y^{n-m} p)$.

Thus, in all cases, we've reduced the degree of one of the two polynomials. By repeatedly applying these latter two, and bottoming out as appropriate, we eventually obtain a quantifier-free equivalent to our formula. □

Corollary 5.6.3. *If F and K are algebraically closed fields such that F is a subfield of K, then $(F, 0, 1, +, \cdot) \preceq (K, 0, 1, +, \cdot)$.*

Proof. Immediate from Proposition 5.5.7. □

Since $\overline{\mathbb{Q}}$ and $\mathbb{C}$ are both algebraically closed, and $(\overline{\mathbb{Q}}, 0, 1, +, \cdot)$ is a substructure of $(\mathbb{C}, 0, 1, +, \cdot)$, we obtain the following corollary.

Corollary 5.6.4. $(\overline{\mathbb{Q}}, 0, 1, +, \cdot) \preceq (\mathbb{C}, 0, 1, +, \cdot)$.

Corollary 5.6.5. *Suppose that F is an algebraically closed field. Every set $X \subseteq F$ that is definable in $(F, 0, 1, +, \cdot)$ is either finite or cofinite.*

Proof. Every atomic formula in one variable is equivalent to either $p(\mathsf{x}) = 0$ or $\neg(p(\mathsf{x}) = 0)$, for some choice of polynomial $p(\mathsf{x})$ in the variable x with integer coefficients. In the former case, notice that any nonzero polynomial has only finitely many roots, so the corresponding definable set is finite. In the latter case, the same argument shows that the corresponding definable set is cofinite. Now notice that the collection of subsets of F that are either finite or cofinite is closed under complement and union. Using Corollary 5.5.6, we conclude that every definable set is either finite or cofinite. □

Corollary 5.6.6. *ACF_0 is complete and ACF_p is complete for all primes p.*

Proof. Apply Proposition 5.5.9, together with the fact that $\mathbb{Q}$ embeds in all fields of characteristic 0, and $\mathbb{Z}/p\mathbb{Z}$ embeds in all fields of characteristic p. □

5.7 Exercises

1. Either prove or give a counterexample: If $\Gamma \subseteq Form_{\mathcal{L}}$ and $\varphi, \psi \in Form_{\mathcal{L}}$, then $\Gamma \vDash \varphi \wedge \psi$ if and only if both $\Gamma \vDash \varphi$ and $\Gamma \vDash \psi$.

2. (a) Let $\mathcal{L} = \{\mathsf{P}\}$ where P is a unary relation symbol. Calculate $I(Cn(\emptyset), n)$ for all $n \in \mathbb{N}^+$.

   (b) Let $\mathcal{L} = \{\mathsf{R}\}$ where R is a binary relation symbol, and let σ be the sentence

   $$(\forall \mathsf{x} \forall \mathsf{y}(\mathsf{Rxy} \rightarrow \mathsf{Ryx})) \wedge \forall \mathsf{x}((\exists \mathsf{y} \mathsf{Rxy}) \rightarrow (\forall \mathsf{y} \mathsf{Rxy})).$$

   Calculate $I(Cn(\sigma), n)$ for all $n \in \mathbb{N}^+$.

   Definition 5.7.1. *A set $A \subseteq \mathbb{N}^+$ is called a* spectrum *if there exists a finite language $\mathcal{L}$ and $\sigma \in Sent_{\mathcal{L}}$ such that $A = Spec(\sigma)$.*

3. (a) Show that every finite set $F \subseteq \mathbb{N}^+$ is a spectrum.

   (b) Show that $\{2n + 1 : n \in \mathbb{N}\}$ is a spectrum.

   (c) Show that $\{n \in \mathbb{N}^+ : n > 1$ and n is composite$\}$ is a spectrum.

4. (a) Show that if $\mathcal{L}$ is a finite language and $\sigma, \tau \in Sent_{\mathcal{L}}$, then $Spec(\sigma) \cup Spec(\tau) = Spec(\sigma \vee \tau)$.

   (b) Show that there exists a finite language $\mathcal{L}$ and $\sigma, \tau \in Sent_{\mathcal{L}}$ such that $Spec(\sigma) \cap Spec(\tau) \neq Spec(\sigma \wedge \tau)$.

   (c) Show that there exists a finite language $\mathcal{L}$ and $\sigma \in Sent_{\mathcal{L}}$ such that $\mathbb{N}^+ \backslash Spec(\sigma) \neq Spec(\neg\sigma)$.

   (d) Show that if $A, B \subseteq \mathbb{N}^+$ are both spectra, then $A \cap B$ is a spectrum.

   Cultural Aside: Although the class of spectra is closed under finite unions and intersections (by parts (a) and (d)), it is an open question whether the class of spectra is closed under complement. This problem is closely tied to problems in complexity theory. As mentioned in class, it turns out that the class of spectra is exactly the collection of subsets of $\mathbb{N}^+$ which are in the complexity class NE, i.e. those accepted by a nondeterministic Turing machine which runs in time $2^{O(n)}$. Thus, the question of whether the class of spectra is closed under complement is equivalent to whether NE = co-NE, the analogue of the question of whether NP = co-NP at a slightly higher complexity level.

5. (**)

   (a) Show that $\{n^2 : n \in \mathbb{N}^+\}$ is a spectrum.

   (b) Show that $\{p \in \mathbb{N}^+ : p$ is prime$\}$ is a spectrum.

6. Using the fact that DLO has QE, determine (with proof) all definable subsets of $\mathbb{Q}^2$ in the structure $(\mathbb{Q}, <)$.

7. Let $\mathcal{L} = \{\mathsf{R}\}$ where R is a binary relation symbol. Consider the $\mathcal{L}$-structure $\mathcal{M}$ that is the linear ordering obtained by putting one copy of $\mathbb{R}$ after another. More formally, $M = (\mathbb{R} \times \{0\}) \cup (\mathbb{R} \times \{1\})$, where we order elements as usual in each copy, and where $(a, 0) < (b, 1)$ for all $a, b \in \mathbb{R}$. Show that $\mathcal{M}$ is not isomorphic to $(\mathbb{R}, <)$.

   Note: If we do the same construction with $\mathbb{Q}$, then the resulting linear ordering is a countable model of DLO, so is isomorphic to $(\mathbb{Q}, <)$.

8. Let $\mathcal{L} = \{\mathsf{0}, \mathsf{1}, +\}$ where $\mathsf{0}$ and $\mathsf{1}$ are constant symbols, and $+$ is a binary function symbol. Let $\mathcal{M}$ be the $\mathcal{L}$-structure where $M = \mathbb{Z}$, and where the symbols are interpreted in the usual way. Let $T = Th(\mathcal{M})$.

(a) Show that if $X \subseteq \mathbb{Z}$ is definable in $\mathcal{M}$ by a quantifier-free formula, then X is either finite or cofinite.

(b) Show that T does not have QE by finding a definable subset of $\mathbb{Z}$ that is neither finite nor cofinite.

9. Let $\mathcal{L} = \{\mathsf{R}\}$ where R is a binary relation symbol. For each $n \in \mathbb{N}^+$, let σ_n be the sentence

$$\forall \mathsf{y} \exists \mathsf{x}_1 \exists \mathsf{x}_2 \cdots \exists \mathsf{x}_\mathsf{n} \Big(\bigwedge_{1 \leq \mathsf{i} < \mathsf{j} \leq \mathsf{n}} \neg(\mathsf{x}_\mathsf{i} = \mathsf{x}_\mathsf{j}) \wedge \bigwedge_{\mathsf{i}=1}^{\mathsf{n}} \mathsf{R}\mathsf{x}_\mathsf{i}\mathsf{y} \Big)$$

and let τ_n be the sentence

$$\exists \mathsf{x}_1 \exists \mathsf{x}_2 \cdots \exists \mathsf{x}_\mathsf{n} \Big(\bigwedge_{1 \leq \mathsf{i} < \mathsf{j} \leq \mathsf{n}} \neg(\mathsf{x}_\mathsf{i} = \mathsf{x}_\mathsf{j}) \wedge \bigwedge_{1 \leq \mathsf{i} < \mathsf{j} \leq \mathsf{n}} \neg \mathsf{R}\mathsf{x}_\mathsf{i}\mathsf{x}_\mathsf{j} \Big)$$

Finally, let

$$\Sigma = \{\forall \mathsf{x}\mathsf{R}\mathsf{x}\mathsf{x}, \forall \mathsf{x} \forall \mathsf{y}(\mathsf{R}\mathsf{x}\mathsf{y} \rightarrow \mathsf{R}\mathsf{y}\mathsf{x}), \forall \mathsf{x} \forall \mathsf{y} \forall \mathsf{z}((\mathsf{R}\mathsf{x}\mathsf{y} \wedge \mathsf{R}\mathsf{y}\mathsf{z}) \rightarrow \mathsf{R}\mathsf{x}\mathsf{z})\} \cup \{\sigma_n : n \in \mathbb{N}^+\} \cup \{\tau_n : n \in \mathbb{N}^+\}$$

and let $T = Cn(\Sigma)$. Notice that models of T are equivalence relations such that there are infinitely many equivalence classes, and such that every equivalence class is infinite.

(a) Show that any two countable models of T are isomorphic, and hence that T is complete.

(b) Let $\mathcal{M}$ be the $\mathcal{L}$-structure where $M = \{n \in \mathbb{N} : n \geq 2\}$ and $\mathsf{R}^\mathcal{M} = \{(a, b) \in M^2 :$ For all primes p, we have $p \mid a$ if and only if $p \mid b\}$. Let $\mathcal{N}$ be the $\mathcal{L}$-structure where $N = \mathbb{R}^2$ and $\mathsf{R}^\mathcal{N} = \{((a_1, b_1), (a_2, b_2)) \in N^2 : a_2 - a_1 = b_2 - b_1\}$. Show that $\mathcal{M} \equiv \mathcal{N}$ and $\mathcal{M} \ncong \mathcal{N}$.

(c) Show that T has QE.

10. Let $\mathcal{L} = \{\mathsf{f}\}$, where f is a unary function symbol. For each $n \in \mathbb{N}^+$, let σ_n be the sentence $\forall \mathsf{x} \neg(\mathsf{f}\mathsf{f} \cdots \mathsf{f}\mathsf{x} = \mathsf{x})$, where there are n many f's. Let

$$\Sigma = \{\forall \mathsf{x} \forall \mathsf{y}(\mathsf{f}\mathsf{x} = \mathsf{f}\mathsf{y} \rightarrow \mathsf{x} = \mathsf{y}), \forall \mathsf{y} \exists \mathsf{x}(\mathsf{f}\mathsf{x} = \mathsf{y})\} \cup \{\sigma_n : n \in \mathbb{N}^+\}.$$

Let $T = Cn(\Sigma)$. Thus, models of T are structures where f is interpreted as a bijection without any finite cycles. For example, the structure $\mathcal{M}$ with universe $M = \mathbb{Z}$ and with $\mathsf{f}^\mathcal{M}(a) = a + 1$ for all $a \in \mathbb{Z}$ is a model of T.

(a) Show that all models of T are infinite.

(b) Give an example of a model of T that is not isomorphic to the example described above.

(c) Show that T has QE.

Chapter 6

Soundness, Completeness, and Compactness

6.1 Syntactic Implication and Soundness

Ever since we defined the syntactic construction of formulas we first-order logic, we have stayed on the semantic side. That is, we talked about structures and variable assignments (which taken together form the analogue of truth assignments), recursively defined truth of a formula in terms of these objects, and then we proceeded to define semantic implication. We now extend the proof rules that we developed in propositional logic in order to define a concept of syntactic implication in first-order logic. Most of our new rules will deal with quantifiers, but we also have the special equality symbol in every first-order language.

Once again, the objects that we will manipulate will be pairs, where the first component is a finite sequence of formulas, and the second is a formula. Given a finite sequence $S \in Form_{\mathcal{L}}^*$ and a formula $\varphi \in Form_{\mathcal{L}}$, we will write $S \vdash \varphi$ to intuitively mean that there is a formal syntactic proof of φ from the assumptions that appear in the sequence S. We begin with the most basic proofs, and we now have two types.

Trivial Implications:

- We can assert $S \vdash \varphi$ if φ appears as an element in the sequence S, i.e. if there exists an $i < |S|$ such that $S(i) = \gamma$. We denote these uses of this by writing $(Assume_{\mathcal{L}})$, since our conclusion appears in our assumptions.
- For any sequence S and any $t \in Term_{\mathcal{L}}$, we can assert $S \vdash t = t$. We denote a use of this rule by writing $(= Refl)$.

With these in hand, we describe ways to generate new formal proof from ones that we already have established. We begin with all of the old rules, but now add five new rules: one dealing with equality, and two for each quantifier:

$$\frac{S \vdash \varphi \wedge \psi}{S \vdash \varphi} \ (\wedge EL) \qquad \frac{S \vdash \varphi \wedge \psi}{S \vdash \psi} \ (\wedge ER) \qquad \frac{S \vdash \varphi \quad S \vdash \psi}{S \vdash \varphi \wedge \psi} \ (\wedge I)$$

$$\frac{S \vdash \varphi}{S \vdash \varphi \vee \psi} \ (\vee IL) \qquad \frac{S \vdash \psi}{S \vdash \varphi \vee \psi} \ (\vee IR)$$

$$\frac{S \vdash \varphi \to \psi}{S, \varphi \vdash \psi} \ (\to E) \qquad \frac{S, \varphi \vdash \psi}{S \vdash \varphi \to \psi} \ (\to I)$$

$$\frac{S,\varphi\vdash\theta \quad S,\psi\vdash\theta}{S,\varphi\vee\psi\vdash\theta}\ (\vee PC) \qquad \frac{S,\psi\vdash\varphi \quad S,\neg\psi\vdash\varphi}{S\vdash\varphi}\ (\neg PC)$$

$$\frac{S,\neg\varphi\vdash\psi \quad S,\neg\varphi\vdash\neg\psi}{S\vdash\varphi}\ (Contr)$$

$$\frac{S\vdash\varphi}{S,\gamma\vdash\varphi}\ (Expand) \qquad \frac{S,\gamma,\gamma\vdash\varphi}{S,\gamma\vdash\varphi}\ (Delete) \qquad \frac{S_1,\gamma_1,\gamma_2,S_2\vdash\varphi}{S_1,\gamma_2,\gamma_1,S_2\vdash\varphi}\ (Reorder)$$

Equality Rules:

$$\frac{S\vdash\varphi^t_{\mathsf{x}} \quad S\vdash t=u}{S\vdash\varphi^u_{\mathsf{x}}}\ \text{if } ValidSubst^t_{\mathsf{x}}(\varphi)=1=ValidSubst^u_{\mathsf{x}}(\varphi)\ \ (=Sub)$$

Existential Rules:

$$\frac{S\vdash\varphi^t_{\mathsf{x}}}{S\vdash\exists\mathsf{x}\varphi}\ \text{if } ValidSubst^t_{\mathsf{x}}(\varphi)=1\ \ (\exists I)$$

$$\frac{S,\varphi^{\mathsf{y}}_{\mathsf{x}}\vdash\psi}{S,\exists\mathsf{x}\varphi\vdash\psi}\ \text{if } \mathsf{y}\notin FreeVar(S,\exists\mathsf{x}\varphi,\psi) \text{ and } ValidSubst^{\mathsf{y}}_{\mathsf{x}}(\varphi)=1\ \ (\exists P)$$

Universal Rules:

$$\frac{S\vdash\forall\mathsf{x}\varphi}{S\vdash\varphi^t_{\mathsf{x}}}\ \text{if } ValidSubst^t_{\mathsf{x}}(\varphi)=1\ \ (\forall E)$$

$$\frac{S\vdash\varphi^{\mathsf{y}}_{\mathsf{x}}}{S\vdash\forall\mathsf{x}\varphi}\ \text{if } \mathsf{y}\notin FreeVar(S,\forall\mathsf{x}\varphi) \text{ and } ValidSubst^{\mathsf{y}}_{\mathsf{x}}(\varphi)=1\ \ (\forall I)$$

To formalize these ideas, we follow the outline from propositional logic.

Definition 6.1.1. *Let $\mathcal{L}$ be language. We define the following:*

- $Line_{\mathcal{L}}=Form^*_{\mathcal{L}}\times Form_{\mathcal{L}}$.
- $Assume_{\mathcal{L}}=\{(S,\varphi)\in Line_{\mathcal{L}}:$ *There exists* $i<|S|$ *such that* $S(i)=\varphi\}$.
- $EqRefl_{\mathcal{L}}=\{(S,t=t):S\in Form^*_{\mathcal{L}},t\in Term_{\mathcal{L}}\}$.

As in propositional logic, we then define a set-valued functions from $Line_{\mathcal{L}}$ (or $Line^2_{\mathcal{L}}$) to $Line_{\mathcal{L}}$ for each rule, and let $\mathcal{H}$ be the collection of all such functions. With this collection of function in hand, we define the following.

Definition 6.1.2. *Let $S\subseteq Form^*_{\mathcal{L}}$ and let $\varphi\in Form_{\mathcal{L}}$. We write $S\vdash\varphi$ to mean that*

$$(S,\varphi)\in(Line_{\mathcal{L}},Assume_{\mathcal{L}}\cup EqRefl_{\mathcal{L}},\mathcal{H}).$$

Definition 6.1.3. *A* deduction *is a witnessing sequence in $(Line_{\mathcal{L}},Assume_{\mathcal{L}}\cup EqRefl_{\mathcal{L}},\mathcal{H})$.*

We have defined the concept of $S\vdash\varphi$ when S is a finite sequence of formulas. Using this, we can define $\Gamma\vdash\varphi$ in the case where Γ is an arbitrary (possibly infinite) set of formulas.

Definition 6.1.4. *Let $\mathcal{L}$ be a language, let $\Gamma\subseteq Form_{\mathcal{L}}$, and let $\varphi\in Form_{\mathcal{L}}$. We write $\Gamma\vdash\varphi$ if there exists a finite sequence $S\in\Gamma^*$ such that $S\vdash\varphi$. We pronounce $\Gamma\vdash\varphi$ as "Γ syntactically implies φ".*

For example, consider the language $\mathcal{L} = \{\mathsf{f}, \mathsf{g}\}$ where f and g are unary function symbols. Given *distinct* $\mathsf{x}, \mathsf{y} \in Var$, here is an example of a deduction showing that $\forall \mathsf{x}(\mathsf{fgx} = \mathsf{x}) \vdash \forall \mathsf{y} \exists \mathsf{x}(\mathsf{fx} = \mathsf{y})$:

$$\begin{aligned}
&\forall \mathsf{x}(\mathsf{fgx} = \mathsf{x}) \vdash \forall \mathsf{x}(\mathsf{fgx} = \mathsf{x}) && (Assume_{\mathcal{L}}) && (1)\\
&\forall \mathsf{x}(\mathsf{fgx} = \mathsf{x}) \vdash \mathsf{fgy} = \mathsf{y} && (\forall E \text{ on } 2 \text{ with } \mathsf{fgx} = \mathsf{x}) && (2)\\
&\forall \mathsf{x}(\mathsf{fgx} = \mathsf{x}) \vdash \exists \mathsf{x}(\mathsf{fx} = \mathsf{y}) && (\exists I \text{ on } 1 \text{ with } \mathsf{fx} = \mathsf{y}) && (3)\\
&\forall \mathsf{x}(\mathsf{fgx} = \mathsf{x}) \vdash \forall \mathsf{y} \exists \mathsf{x}(\mathsf{fx} = \mathsf{y}) && (\forall I \text{ on } 3 \text{ with } \exists \mathsf{x}(\mathsf{fx} = \mathsf{y})) && (4)
\end{aligned}$$

Notice that this deduction is a formal syntactic derivation of the fact that if f is a left inverse of g (where f and g are functions with the same common domain and codomain), then f is surjective. Furthermore, we are using the fact that $ValidSubst^{\mathsf{y}}_{\mathsf{x}}(\mathsf{fgx} = \mathsf{x}) = 1$ on line (2), that $ValidSubst^{\mathsf{gx}}_{\mathsf{x}}(\mathsf{fx} = \mathsf{y}) = 1$ on line (3), and that both $\mathsf{y} \notin FreeVar(\forall \mathsf{x}(\mathsf{fgx} = \mathsf{x}), \forall \mathsf{y} \exists \mathsf{x}(\mathsf{fx} = \mathsf{y}))$ and $ValidSubst^{\mathsf{y}}_{\mathsf{y}}(\exists \mathsf{x}(\mathsf{fx} = \mathsf{y})) = 1$ on line (4).

For another example in the same language, given distinct $\mathsf{x}, \mathsf{y} \in Var$, here is a deduction showing that $\forall \mathsf{x}(\mathsf{fgx} = \mathsf{x}) \vdash \forall \mathsf{x} \forall \mathsf{y}(\mathsf{gx} = \mathsf{gy} \to \mathsf{x} = \mathsf{y})$:

$$\begin{aligned}
&\forall \mathsf{x}(\mathsf{fgx} = \mathsf{x}), \mathsf{gx} = \mathsf{gy} \vdash \forall \mathsf{x}(\mathsf{fgx} = \mathsf{x}) && (Assume_{\mathcal{L}}) && (1)\\
&\forall \mathsf{x}(\mathsf{fgx} = \mathsf{x}), \mathsf{gx} = \mathsf{gy} \vdash \mathsf{fgx} = \mathsf{x} && (\forall E \text{ on } 1 \text{ with } \mathsf{fgx} = \mathsf{x}) && (2)\\
&\forall \mathsf{x}(\mathsf{fgx} = \mathsf{x}), \mathsf{gx} = \mathsf{gy} \vdash \mathsf{fgy} = \mathsf{y} && (\forall E \text{ on } 1 \text{ with } \mathsf{fgx} = \mathsf{x}) && (3)\\
&\forall \mathsf{x}(\mathsf{fgx} = \mathsf{x}), \mathsf{gx} = \mathsf{gy} \vdash \mathsf{gx} = \mathsf{gy} && (Assume_{\mathcal{L}}) && (4)\\
&\forall \mathsf{x}(\mathsf{fgx} = \mathsf{x}), \mathsf{gx} = \mathsf{gy} \vdash \mathsf{fgx} = \mathsf{fgx} && (= Refl) && (5)\\
&\forall \mathsf{x}(\mathsf{fgx} = \mathsf{x}), \mathsf{gx} = \mathsf{gy} \vdash \mathsf{fgx} = \mathsf{fgy} && (= Sub \text{ on } 4 \text{ and } 5 \text{ with } \mathsf{fgx} = \mathsf{fz}) && (6)\\
&\forall \mathsf{x}(\mathsf{fgx} = \mathsf{x}), \mathsf{gx} = \mathsf{gy} \vdash \mathsf{x} = \mathsf{fgy} && (= Sub \text{ on } 2 \text{ and } 6 \text{ with } \mathsf{z} = \mathsf{fgy}) && (7)\\
&\forall \mathsf{x}(\mathsf{fgx} = \mathsf{x}), \mathsf{gx} = \mathsf{gy} \vdash \mathsf{x} = \mathsf{y} && (= Sub \text{ on } 3 \text{ and } 7 \text{ with } \mathsf{x} = \mathsf{z}) && (8)\\
&\forall \mathsf{x}(\mathsf{fgx} = \mathsf{x}) \vdash \mathsf{gx} = \mathsf{gy} \to \mathsf{x} = \mathsf{y} && (\to I \text{ on } 8) && (9)\\
&\forall \mathsf{x}(\mathsf{fgx} = \mathsf{x}) \vdash \forall \mathsf{y}(\mathsf{gx} = \mathsf{gy} \to \mathsf{x} = \mathsf{y}) && (\forall I \text{ on } 9 \text{ with } \mathsf{gx} = \mathsf{gy} \to \mathsf{x} = \mathsf{y}) && (10)\\
&\forall \mathsf{x}(\mathsf{fgx} = \mathsf{x}) \vdash \forall \mathsf{x} \forall \mathsf{y}(\mathsf{gx} = \mathsf{gy} \to \mathsf{x} = \mathsf{y}) && (\forall I \text{ on } 10 \text{ with } \forall \mathsf{y}(\mathsf{gx} = \mathsf{gy} \to \mathsf{x} = \mathsf{y})) && (11)
\end{aligned}$$

We leave it as an exercise to check that all of the restrictions on the rules are satisfied (i.e. that $ValidSubst$ equals 1 in all appropriate cases, and no variable is free in the wrong circumstance).

We now prove a few simple results that will be essential later.

Proposition 6.1.5. *For any $t, u \in Term_{\mathcal{L}}$, we have $t = u \vdash u = t$.*

Proof. Let $t, u \in Term_{\mathcal{L}}$ be arbitrary. Consider the following deduction:

$$\begin{aligned}
&t = u \vdash t = t && (= Refl) && (1)\\
&t = u \vdash t = u && (Assume_{\mathcal{L}}) && (2)\\
&t = u \vdash u = t && (= Sub \text{ on } 1 \text{ and } 2 \text{ with } \mathsf{x} = t) && (3)
\end{aligned}$$

□

Proposition 6.1.6. *For any $t, u, w \in Term_{\mathcal{L}}$, we have $t = u, u = w \vdash t = w$.*

Proof. Let $t, u, w \in Term_{\mathcal{L}}$ be arbitrary.

$$\begin{aligned}
&t = u, u = w \vdash t = u && (Assume_{\mathcal{L}}) && (1)\\
&t = u, u = w \vdash u = w && (Assume_{\mathcal{L}}) && (2)\\
&t = u, u = w \vdash t = w && (= Sub \text{ on } 1 \text{ and } 2 \text{ with } t = \mathsf{x}) && (3)
\end{aligned}$$

□

Proposition 6.1.7. *For any* $\mathsf{R} \in \mathcal{R}_k$ *and any* $t_1, t_2, \ldots, t_k \in Term_{\mathcal{L}}$*, we have*

$$\{\mathsf{R}t_1t_2\cdots t_k, t_1 = u_1, t_2 = u_2, \ldots, t_k = u_k\} \vdash \mathsf{R}u_1u_2\cdots u_k.$$

Proof. Let $\mathsf{R} \in \mathcal{R}_k$ and $t_1, t_2, \ldots, t_k \in Term_{\mathcal{L}}$ be arbitrary. Since Var is infinite and each term has only finitely many variables that occur in it, we can fix $\mathsf{x} \notin \bigcup_{i=1}^{k}(OccurVar(t_i) \cup OccurVar(u_i))$. Let S be the sequence of formulas $\mathsf{R}t_1t_2\cdots t_k, t_1 = u_1, t_2 = u_2, \ldots, t_k = u_k$. Consider the following deduction:

$S \vdash \mathsf{R}t_1t_2\cdots t_k$	$(Assume_{\mathcal{L}})$	(1)
$S \vdash t_1 = u_1$	$(Assume_{\mathcal{L}})$	(2)
$S \vdash \mathsf{R}u_1t_2t_3\cdots t_k$	$(= Sub$ on 1 and 2 with $\mathsf{R}\mathsf{x}t_2t_3\cdots t_k)$	(3)
$S \vdash t_2 = u_2$	$(Assume_{\mathcal{L}})$	(4)
$S \vdash \mathsf{R}u_1u_2t_3\cdots t_k$	$(= Sub$ on 3 and 4 with $\mathsf{R}u_1\mathsf{x}t_3\cdots t_k)$	(5)
$\vdots$		
$S \vdash t_k = u_k$	$(Assume_{\mathcal{L}})$	$(2k)$
$S \vdash \mathsf{R}u_1u_2\cdots u_k$	$(= Sub$ on $2k-1$ and $2k$ with $\mathsf{R}u_1u_2\cdots\mathsf{x})$	$(2k+1)$

□

Proposition 6.1.8. *For any* $\mathsf{f} \in \mathcal{F}_k$ *and any* $t_1, t_2, \ldots, t_k \in Term_{\mathcal{L}}$*, we have*

$$\{t_1 = u_2, t_2 = u_2, \ldots, t_k = u_k\} \vdash \mathsf{f}t_1t_2\cdots t_k = \mathsf{f}u_1u_2\cdots u_k$$

Proof. Let $\mathsf{f} \in \mathcal{F}_k$ and $t_1, t_2, \ldots, t_k \in Term_{\mathcal{L}}$ be arbitrary. Since Var is infinite and each term has only finitely many variables that occur in it, we can fix $\mathsf{x} \notin \bigcup_{i=1}^{k}(OccurVar(t_i) \cup OccurVar(u_i))$. Let S be the sequence of formulas $t_1 = u_1, t_2 = u_2, \ldots, t_k = u_k$. Consider the following deduction:

$S \vdash \mathsf{f}t_1t_2\cdots t_k = \mathsf{f}t_1t_2\cdots t_k$	$(= Refl)$	(1)
$S \vdash t_1 = u_1$	$(Assume_{\mathcal{L}})$	(2)
$S \vdash \mathsf{f}t_1t_2\cdots t_k = \mathsf{f}u_1t_2\cdots t_k$	$(= Sub$ on 1 and 2 with $\mathsf{f}t_1t_2\cdots t_k = \mathsf{f}\mathsf{x}t_2\cdots t_k)$	(3)
$S \vdash t_2 = u_2$	$(Assume_{\mathcal{L}})$	(4)
$S \vdash \mathsf{f}t_1t_2\cdots t_k = \mathsf{f}u_1u_2\cdots t_k$	$(= Sub$ on 1 and 2 with $\mathsf{f}t_1t_2\cdots t_k = \mathsf{f}u_1\mathsf{x}\cdots t_k)$	(3)
$\vdots$		
$S \vdash t_k = u_k$	$(Assume_{\mathcal{L}})$	$(2k)$
$S \vdash \mathsf{f}t_1t_2\cdots t_k = \mathsf{f}u_1u_2\cdots u_k$	$(= Sub$ on $2k-1$ and $2k$ with $\mathsf{f}t_1t_2\cdots t_k = \mathsf{f}u_1u_2\cdots\mathsf{x})$	$(2k+1)$

□

Proposition 6.1.9. *For any* $\varphi \in Form_{\mathcal{L}}$ *and* $\mathsf{x} \in Var$*, we have* $\exists\mathsf{x}\varphi \vdash \neg\forall\mathsf{x}\neg\varphi$*.*

Proof. Let $\varphi \in Form_{\mathcal{L}}$ and $\mathsf{x} \in Var$ be arbitrary. Since Var is infinite and each formula has only finitely many variables that occur in it, we can fix $\mathsf{y} \in Var$ with both $\mathsf{y} \neq \mathsf{x}$ with $\mathsf{y} \notin OccurVar(\varphi)$. Consider the

following deduction:

$\varphi^{\mathsf{y}}_{\mathsf{x}}, \neg\neg\forall\mathsf{x}\neg\varphi, \neg\forall\mathsf{x}\neg\varphi \vdash \neg\forall\mathsf{x}\neg\varphi$	$(Assume_{\mathcal{L}})$	(1)
$\varphi^{\mathsf{y}}_{\mathsf{x}}, \neg\neg\forall\mathsf{x}\neg\varphi, \neg\forall\mathsf{x}\neg\varphi \vdash \neg\neg\forall\mathsf{x}\neg\varphi$	$(Assume_{\mathcal{L}})$	(2)
$\varphi^{\mathsf{y}}_{\mathsf{x}}, \neg\neg\forall\mathsf{x}\neg\varphi \vdash \forall\mathsf{x}\neg\varphi$	($Contr$ on 1 and 2)	(3)
$\varphi^{\mathsf{y}}_{\mathsf{x}}, \neg\neg\forall\mathsf{x}\neg\varphi \vdash \neg\varphi^{\mathsf{y}}_{\mathsf{x}}$	($\forall E$ on 3)	(4)
$\varphi^{\mathsf{y}}_{\mathsf{x}}, \neg\neg\forall\mathsf{x}\neg\varphi \vdash \varphi^{\mathsf{y}}_{\mathsf{x}}$	$(Assume_{\mathcal{L}})$	(5)
$\varphi^{\mathsf{y}}_{\mathsf{x}} \vdash \neg\forall\mathsf{x}\neg\varphi$	($Contr$ on 4 and 5)	(6)
$\exists\mathsf{x}\varphi \vdash \neg\forall\mathsf{x}\neg\varphi$	($\exists P$ on 6)	(7)

□

Proposition 6.1.10. *For any $\varphi \in Form_{\mathcal{L}}$ and $\mathsf{x} \in Var$, we have $\neg\exists\mathsf{x}\neg\varphi \vdash \forall\mathsf{x}\varphi$.*

Proof. Let $\varphi \in Form_{\mathcal{L}}$ and $\mathsf{x} \in Var$ be arbitrary. Since Var is infinite and each formula has only finitely many variables that occur in it, we can fix $\mathsf{y} \in Var$ with both $\mathsf{y} \neq \mathsf{x}$ with $\mathsf{y} \notin OccurVar(\varphi)$. Consider the following deduction:

$\neg\exists\mathsf{x}\neg\varphi, \neg\varphi^{\mathsf{y}}_{\mathsf{x}} \vdash \neg\exists\mathsf{x}\neg\varphi$	$(Assume_{\mathcal{L}})$	(1)
$\neg\exists\mathsf{x}\neg\varphi, \neg\varphi^{\mathsf{y}}_{\mathsf{x}} \vdash \neg\varphi^{\mathsf{y}}_{\mathsf{x}}$	$(Assume_{\mathcal{L}})$	(2)
$\neg\exists\mathsf{x}\neg\varphi, \neg\varphi^{\mathsf{y}}_{\mathsf{x}} \vdash \exists\mathsf{x}\neg\varphi$	($\exists I$ on 2)	(3)
$\neg\exists\mathsf{x}\neg\varphi \vdash \varphi^{\mathsf{y}}_{\mathsf{x}}$	($Contr$ on 1 and 3)	(4)
$\neg\exists\mathsf{x}\neg\varphi \vdash \forall\mathsf{x}\varphi$	($\forall I$ on 4)	(5)

□

The following definition and results following in precisely the same way as they did for propositional logic (because we still include all of the old proof rules).

Definition 6.1.11. Γ *is* inconsistent *if there exists $\theta \in Form_{\mathcal{L}}$ such that $\Gamma \vdash \theta$ and $\Gamma \vdash \neg\theta$. Otherwise, we say that Γ is* consistent*.*

Proposition 6.1.12. *Let $\Gamma_1 \subseteq Form_{\mathcal{L}}$ and $\Gamma_2 \subseteq Form_{\mathcal{L}}$ be such that $\Gamma_1 \subseteq \Gamma_2$. If $\varphi \in Form_{\mathcal{L}}$ is such that $\Gamma_1 \vdash \varphi$, then $\Gamma_2 \vdash \varphi$.*

Proof. See the proof of Proposition 3.4.8. □

Proposition 6.1.13. *Suppose that $S \in Form^*_{\mathcal{L}}$ and $\varphi \in Form_{\mathcal{L}}$ are such that $S \vdash \varphi$. If T is any permutation of S, then $T \vdash \varphi$.*

Proof. See the proof of Proposition 3.4.9. □

Proposition 6.1.14. *If Γ is inconsistent, then $\Gamma \vdash \varphi$ for all $\varphi \in Form_{\mathcal{L}}$.*

Proof. See the proof of Proposition 3.4.10. □

Proposition 6.1.15. *Let $\Gamma \subseteq Form_{\mathcal{L}}$ and let $\varphi \in Form_{\mathcal{L}}$.*

1. *If $\Gamma \cup \{\varphi\}$ is inconsistent, then $\Gamma \vdash \neg\varphi$.*
2. *If $\Gamma \cup \{\neg\varphi\}$ is inconsistent, then $\Gamma \vdash \varphi$.*

Proof. See the proof of Proposition 3.4.11. □

Corollary 6.1.16. *If $\Gamma \subseteq Form_{\mathcal{L}}$ is consistent and $\varphi \in Form_{\mathcal{L}}$, then either $\Gamma \cup \{\varphi\}$ is consistent or $\Gamma \cup \{\neg\varphi\}$ is consistent.*

Proof. See the proof of Corollary 3.4.12. □

Proposition 6.1.17. *Let $\Gamma \subseteq Form_{\mathcal{L}}$ and let $\varphi \in Form_{\mathcal{L}}$.*

1. *If $\Gamma \vdash \varphi$ and $\Gamma \cup \{\varphi\} \vdash \psi$, then $\Gamma \vdash \psi$.*
2. *If $\Gamma \vdash \varphi$ and $\Gamma \vdash \varphi \to \psi$, then $\Gamma \vdash \psi$.*

Proof. See the proof of Proposition 3.4.13. □

Proposition 6.1.18. *$\Gamma \vdash \varphi$ if and only if there is a finite $\Gamma_0 \subseteq \Gamma$ such that $\Gamma_0 \vdash \varphi$.*

Proof. See the proof of Proposition 3.4.14. □

Corollary 6.1.19. *If every finite subset of Γ is consistent, then Γ is consistent.*

Proof. See the proof of Corollary 3.4.15. □

Theorem 6.1.20 (Soundness Theorem)**.**

1. *If $\Gamma \vdash \varphi$, then $\Gamma \vDash \varphi$.*
2. *Every satisfiable set of formulas is consistent.*

Proof.

1. As in the proof of the Soundness Theorem for propositional logic, we prove the following fact: If $S \in Form_{\mathcal{L}}^*$ and $\varphi \in Form_{\mathcal{L}}$ are such that $S \vdash \varphi$, then $S \vDash \varphi$. To see why this suffices, suppose that $\Gamma \vdash \varphi$. By definition, we can then fix $S \in \Gamma^*$ with $S \vdash \varphi$. From here we can conclude that $S \vDash \varphi$. Since every element of S is an element of Γ, it follows that $\Gamma \vDash \varphi$.

   We now prove the statement "Whenever $S \vdash \varphi$, we have $S \vDash \varphi$" by induction. In other words, if G is the set generated by starting with $Assume_{\mathcal{L}} \cup EqRefl_{\mathcal{L}}$ and using our proof rules, and we let

   $$X = \{(S, \varphi) \in G : S \vDash \varphi\},$$

   then we show by induction on G that $X = G$. We begin by noting that if φ appears in the sequence S, then we trivially have $S \vDash \varphi$ by definition. Therefore, $(S, \varphi) \in X$ for all $(S, \varphi) \in Assume_{\mathcal{L}}$. Also, for any $S \in Form_{\mathcal{L}}^*$ and any $t \in Term_{\mathcal{L}}$, we have $S \vDash t = t$ because for any any model $(\mathcal{M}, s)$ of S we trivially have $\overline{s}(t) = \overline{s}(t)$, hence $(\mathcal{M}, s) \vDash t = t$. Therefore, $(S, t = t) \in X$ for all $S \in Form_{\mathcal{L}}^*$ and $t \in Term_{\mathcal{L}}$.

   We now handle the inductive steps. All of the old rules go through in a similar manner as before.

   - We first handle the $= Sub$ rule. Suppose that $S \vDash \varphi_{\mathsf{x}}^t$, that $S \vDash t = u$, and that $ValidSubst_{\mathsf{x}}^t(\varphi) = 1 = ValidSubst_{\mathsf{x}}^u(\varphi)$. We need to show that $S \vDash \varphi_{\mathsf{x}}^u$. Let $(\mathcal{M}, s)$ be an arbitrary model of S. Since $S \vDash \varphi_{\mathsf{x}}^t$, we have that $(\mathcal{M}, s) \vDash \varphi_{\mathsf{x}}^t$. Also, since $S \vDash t = u$, we have that $(\mathcal{M}, s) \vDash t = u$, and hence $\overline{s}(t) = \overline{s}(u)$. Since

     $$(\mathcal{M}, s) \vDash \varphi_{\mathsf{x}}^t$$

     we can use Theorem 4.6.7 together with the fact that $ValidSubst_{\mathsf{x}}^t(\varphi) = 1$ to conclude that

     $$(\mathcal{M}, s[\mathsf{x} \Rightarrow \overline{s}(t)]) \vDash \varphi.$$

Since $\overline{s}(t) = \overline{s}(u)$, it follows that

$$(\mathcal{M}, s[\mathsf{x} \Rightarrow \overline{s}(u)]) \vDash \varphi.$$

Finally, we can use Theorem 4.6.7 together with the fact that $ValidSubst_{\mathsf{x}}^{u}(\varphi) = 1$ to conclude that

$$(\mathcal{M}, s) \vDash \varphi_{\mathsf{x}}^{u}.$$

Since $(\mathcal{M}, s)$ was an arbitrary model of S, it follows that $S \vDash \varphi_{\mathsf{x}}^{u}$.

- We now handle the $\exists I$ rule. Suppose that $S \vDash \varphi_{\mathsf{x}}^{t}$ where $ValidSubst_{\mathsf{x}}^{t}(\varphi) = 1$. We need to show that $S \vDash \exists \mathsf{x}\varphi$. Let $(\mathcal{M}, s)$ be an arbitrary model of S. Since $S \vDash \varphi_{\mathsf{x}}^{t}$, it follows that $(\mathcal{M}, s) \vDash \varphi_{\mathsf{x}}^{t}$. Using Theorem 4.6.7 together with the fact that $ValidSubst_{\mathsf{x}}^{t}(\varphi) = 1$, we have

  $$(\mathcal{M}, s[\mathsf{x} \Rightarrow \overline{s}(t)]) \vDash \varphi.$$

  Therefore, there exists $a \in M$ such that

  $$(\mathcal{M}, s[\mathsf{x} \Rightarrow a]) \vDash \varphi,$$

  from which we conclude that

  $$(\mathcal{M}, s) \vDash \exists \mathsf{x}\varphi.$$

  Since $(\mathcal{M}, s)$ was an arbitrary model of S, it follows that $S \vDash \exists \mathsf{x}\varphi$.

- Let's next attack the $\exists P$ rule. Suppose that $S, \varphi_{\mathsf{x}}^{\mathsf{y}} \vDash \psi$, that $\mathsf{y} \notin FreeVar(S, \exists \mathsf{x}\varphi, \psi)$, and that $ValidSubst_{\mathsf{x}}^{\mathsf{y}}(\varphi) = 1$. We need to show that $S, \exists \mathsf{x}\varphi \vDash \psi$. Let $(\mathcal{M}, s)$ be an arbitrary model of $S, \exists \mathsf{x}\varphi$. Since $(\mathcal{M}, s) \vDash \exists \mathsf{x}\varphi$, we may fix $a \in M$ such that $(\mathcal{M}, s[\mathsf{x} \Rightarrow a]) \vDash \varphi$. We first divide into two cases to show that $(\mathcal{M}, s[\mathsf{y} \Rightarrow a]) \vDash \varphi_{\mathsf{x}}^{\mathsf{y}}$.
  - *Case 1:* Suppose that $\mathsf{y} = \mathsf{x}$. We then have $\varphi_{\mathsf{x}}^{\mathsf{y}} = \varphi_{\mathsf{x}}^{\mathsf{x}} = \varphi$ and $s[\mathsf{x} \Rightarrow a] = s[\mathsf{y} \Rightarrow a]$, hence $(\mathcal{M}, s[\mathsf{y} \Rightarrow a]) \vDash \varphi_{\mathsf{x}}^{\mathsf{y}}$ because $(\mathcal{M}, s[\mathsf{x} \Rightarrow a]) \vDash \varphi$.
  - *Case 2:* Suppose that $\mathsf{y} \neq \mathsf{x}$. We know that $(\mathcal{M}, s[\mathsf{x} \Rightarrow a]) \vDash \varphi$, so since $\mathsf{y} \neq \mathsf{x}$ and $\mathsf{y} \notin FreeVar(\varphi)$, we can conclude that

    $$(\mathcal{M}, (s[\mathsf{y} \Rightarrow a])[\mathsf{x} \Rightarrow a]) \vDash \varphi.$$

    From here, it follows that

    $$(\mathcal{M}, (s[\mathsf{y} \Rightarrow a])[\mathsf{x} \Rightarrow \overline{s[\mathsf{y} \Rightarrow a]}(\mathsf{y})]) \vDash \varphi,$$

    so using Theorem 4.6.7 together with the fact that $ValidSubst_{\mathsf{x}}^{\mathsf{y}}(\varphi) = 1$, we conclude that

    $$(\mathcal{M}, s[\mathsf{y} \Rightarrow a]) \vDash \varphi_{\mathsf{x}}^{\mathsf{y}}.$$

  Thus, $(\mathcal{M}, s[\mathsf{y} \Rightarrow a]) \vDash \varphi_{\mathsf{x}}^{\mathsf{y}}$ in either case. Now since $(\mathcal{M}, s) \vDash \gamma$ for all $\gamma \in S$ and $\mathsf{y} \notin FreeVar(S)$, we have $(\mathcal{M}, s[\mathsf{y} \Rightarrow a]) \vDash \gamma$ for all $\gamma \in S$. Since we are assuming that $S, \varphi_{\mathsf{x}}^{\mathsf{y}} \vDash \psi$, and we know that $(\mathcal{M}, s[\mathsf{y} \Rightarrow a]) \vDash \gamma$ for all $\gamma \in S$, and that $(\mathcal{M}, s[\mathsf{y} \Rightarrow a]) \vDash \varphi_{\mathsf{x}}^{\mathsf{y}}$, we conclude that $(\mathcal{M}, s[\mathsf{y} \Rightarrow a]) \vDash \psi$. Finally, since $\mathsf{y} \notin FreeVar(\psi)$, it follows that $(\mathcal{M}, s) \vDash \psi$.

- We next do the $\forall E$ rule. Suppose that $S \vDash \forall \mathsf{x}\varphi$ and that $t \in Term_{\mathcal{L}}$ is such that $ValidSubst_{\mathsf{x}}^{t}(\varphi) = 1$. We need to show that $S \vDash \varphi_{\mathsf{x}}^{t}$. Let $(\mathcal{M}, s)$ be an arbitrary model of S. Since $S \vDash \forall x\varphi$, it follows that that $(\mathcal{M}, s) \vDash \forall x\varphi$. By definition, we conclude that $(\mathcal{M}, s[\mathsf{x} \Rightarrow a]) \vDash \varphi$ for all $a \in M$. Now $\overline{s}(t) \in M$, so in particular we have that

  $$(\mathcal{M}, s[\mathsf{x} \Rightarrow \overline{s}(t)]) \vDash \varphi$$

  Using Theorem 4.6.7) together with the fact that $ValidSubst_{\mathsf{x}}^{t}(\varphi) = 1$, it follows that

  $$(\mathcal{M}, s) \vDash \varphi_{\mathsf{x}}^{t}.$$

- We finally end with the $\forall I$ rule. Suppose that $S \vDash \varphi^{\mathsf{y}}_{\mathsf{x}}$, that $\mathsf{y} \notin FreeVar(S, \forall \mathsf{x}\varphi)$, and that $ValidSubst^{\mathsf{y}}_{\mathsf{x}}(\varphi) = 1$. We need to show that $S \vDash \forall \mathsf{x}\varphi$. Let $(\mathcal{M}, s)$ be an arbitrary model of S. We handle two cases.
  - *Case 1:* Suppose that $\mathsf{y} = \mathsf{x}$. Since $\mathsf{y} = \mathsf{x}$, we have $\varphi^{\mathsf{y}}_{\mathsf{x}} = \varphi^{\mathsf{x}}_{\mathsf{x}} = \varphi$. Let $a \in M$ be arbitrary. Since $(\mathcal{M}, s)$ is a model of S, we know that $(\mathcal{M}, s) \vDash \gamma$ for all $\gamma \in S$. Now we are assuming that $\mathsf{x} = \mathsf{y} \notin FreeVar(S)$, so we may conclude that $(\mathcal{M}, s[\mathsf{x} \Rightarrow a]) \vDash \gamma$ for all $\gamma \in S$. Since we are also assuming that $S \vDash \varphi^{\mathsf{y}}_{\mathsf{x}}$, it follows that $(\mathcal{M}, s[\mathsf{x} \Rightarrow a]) \vDash \varphi^{\mathsf{y}}_{\mathsf{x}}$, and hence $(\mathcal{M}, s[\mathsf{x} \Rightarrow a]) \vDash \varphi$ (because $\varphi^{\mathsf{y}}_{\mathsf{x}} = \varphi$ in this case). Now $a \in \mathcal{M}$ was arbitrary, so $(\mathcal{M}, s[\mathsf{x} \Rightarrow a]) \vDash \varphi$ for every $a \in M$. Therefore, by definition, we have $(\mathcal{M}, s) \vDash \forall \mathsf{x}\varphi$.
  - *Case 2:* Suppose that $\mathsf{y} \neq \mathsf{x}$. Let $a \in M$ be arbitrary. Since $(\mathcal{M}, s)$ is a model of S, we know that $(\mathcal{M}, s) \vDash \gamma$ for all $\gamma \in S$. Now we are assuming that $\mathsf{y} \notin FreeVar(S)$, so we may conclude that $(\mathcal{M}, s[\mathsf{y} \Rightarrow a]) \vDash \gamma$ for all $\gamma \in S$. Since we are also assuming that $S \vDash \varphi^{\mathsf{y}}_{\mathsf{x}}$, it follows that $(\mathcal{M}, s[\mathsf{y} \Rightarrow a]) \vDash \varphi^{\mathsf{y}}_{\mathsf{x}}$. Using Theorem 4.6.7 together with the fact $ValidSubst^{\mathsf{y}}_{\mathsf{x}}(\varphi) = 1$, we have
    $$(\mathcal{M}, (s[\mathsf{y} \Rightarrow a])[\mathsf{x} \Rightarrow \overline{s[\mathsf{y} \Rightarrow a]}(\mathsf{y})]) \vDash \varphi,$$
    and hence
    $$(\mathcal{M}, (s[\mathsf{y} \Rightarrow a])[\mathsf{x} \Rightarrow a]) \vDash \varphi.$$
    Since we are assuming that $\mathsf{y} \notin FreeVar(\varphi)$ and $\mathsf{y} \neq \mathsf{x}$, it follows that
    $$(\mathcal{M}, s[\mathsf{x} \Rightarrow a]) \vDash \varphi.$$
    Now $a \in M$ was arbitrary, so $(\mathcal{M}, s[\mathsf{x} \Rightarrow a]) \vDash \varphi$ for every $a \in M$, hence $(\mathcal{M}, s) \vDash \forall \mathsf{x}\varphi$.

  The result follows by induction.

2. Let Γ be an arbitrary satisfiable set of formulas. Fix a model $(\mathcal{M}, s)$ of Γ. Suppose that Γ is inconsistent, and fix $\theta \in Form_{\mathcal{L}}$ such that $\Gamma \vdash \theta$ and $\Gamma \vdash \neg\theta$. We then have $\Gamma \vDash \theta$ and $\Gamma \vDash \neg\theta$ by part (1), hence $(\mathcal{M}, s) \vDash \theta$ and $(\mathcal{M}, s) \vDash \neg\theta$, a contradiction. It follows that Γ is consistent.

□

6.2 Completeness

Let's recall the outline of our proof of the Completeness Theorem for propositional logic. We wanted to show that every consistent set was satisfiable. Suppose then that we had an arbitrary consistent set Γ. Since Γ could consist of many very complex formulas, and perhaps no simple formulas, it seemed hard to define an appropriate truth assignment. Thus, our first step was to enlarge Γ to a consistent and complete set Δ. In particular, we then had that property that for every $\mathsf{A} \in P$, either $\mathsf{A} \in \Delta$ or $\neg\mathsf{A} \in \Delta$. With this start, we were able to define an appropriate truth assignment M, and then continue to use the fact that Δ was complete to verify that $v_M(\delta) = 1$ for all $\delta \in \Delta$.

Suppose now that we are in the first-order logic setting. Thus, we have a language $\mathcal{L}$ and a set $\Gamma \subseteq Form_{\mathcal{L}}$ that is consistent. We will take our cue from propositional logic, and first expand the set appropriately. Here is the definition.

Definition 6.2.1. *Suppose that $\mathcal{L}$ is a language and that $\Delta \subseteq Form_{\mathcal{L}}$. We say that Δ is* complete *if for all $\varphi \in Form_{\mathcal{L}}$, either $\varphi \in \Delta$ or $\neg\varphi \in \Delta$.*

Notice that this definition resembles the definition of a complete theory, but differs in the fact that we are assuming that either $\varphi \in \Delta$ or $\neg\varphi \in \Delta$ for each *formula* φ (which is more general than for each *sentence*). Let's assume that we can indeed expand every consistent set Γ to a set Δ that is both consistent

and complete (the proof is completely analogous to the proof in the propositional logic case). In order to show that Γ is satisfiable, it suffices to show that Δ is satisfiable. Thus, we need to construct an $\mathcal{L}$-structure $\mathcal{M}$ and a variable assignment $s\colon Var \to M$ such that $(\mathcal{M}, s) \vDash \delta$ for all $\delta \in \Delta$. Now all that we have is the syntactic information that Δ provides, so it seems that the only way to proceed is to define our $\mathcal{M}$ from these syntactic objects. Since terms intuitively name elements, it is natural to try to define the universe M to simply be $Term_{\mathcal{L}}$. We would then define the structure as follows:

1. $\mathsf{c}^{\mathcal{M}} = \mathsf{c}$ for all $\mathsf{c} \in \mathcal{C}$.
2. $\mathsf{R}^{\mathcal{M}} = \{(t_1, t_2, \ldots, t_k) \in M^k : \mathsf{R}t_1t_2 \ldots t_k \in \Delta\}$ for all $\mathsf{R} \in \mathcal{R}_k$.
3. $\mathsf{f}^{\mathcal{M}}(t_1, t_2, \ldots, t_k) = \mathsf{f}t_1t_2 \cdots t_k$ for all $\mathsf{f} \in \mathcal{F}_k$ and all $t_1, t_2, \ldots, t_k \in M$.

Finally, it seems reasonable to define $s\colon Var \to M$ to be the variable assignment $s(\mathsf{x}) = \mathsf{x}$ for all $\mathsf{x} \in Var$.

Despite the promise and elegance of this approach, there is one minor problem and one major problem that we must address. First, let's think about the minor problem. Suppose that $\mathcal{L} = \{\mathsf{f}, \mathsf{e}\}$ is the basic group theory language, and that Γ is the set of group axioms. Suppose that $\Delta \supseteq \Gamma$ is consistent and complete. We then have $\mathsf{fee} = \mathsf{e} \in \Delta$ because $\Gamma \vdash \mathsf{fee} = \mathsf{e}$. However, the two terms fee and e are syntactically different objects. In other words, if we follow the above idea by letting $M = Term_{\mathcal{L}}$, we would run into a problem because fee and e are distinct, despite the fact that Δ says that they must be equal. Of course, when we have distinct objects that we want to make equal, we should define an equivalence relation. The natural relation here is to define $\sim$ on $Term_{\mathcal{L}}$ by letting $t \sim u$ mean that $t = u \in \Delta$. We would then need to check that $\sim$ is an equivalence relation and that the definition of the structure above is independent of our choice of representatives for the classes. This is all fairly straightforward, and we will carry the details below.

On to the more serious obstacle. Suppose that $\mathcal{L} = \{\mathsf{P}\}$ where P is a unary relation symbol. Let $\Gamma = \{\neg\mathsf{P}\mathsf{x} : \mathsf{x} \in Var\} \cup \{\neg(\mathsf{x} = \mathsf{y}) : \mathsf{x}, \mathsf{y} \in Var \text{ with } \mathsf{x} \neq \mathsf{y}\} \cup \{\exists\mathsf{x}\mathsf{P}\mathsf{x}\}$ and notice that Γ is consistent because it is satisfiable (let $M = \mathbb{N}$, let $s\colon Var \to \mathbb{N}$ be $s(\mathsf{x}_k) = k + 1$, and let $\mathsf{P}^{\mathcal{M}} = \{0\}$). Suppose that $\Delta \supseteq \Gamma$ is consistent and complete. In the structure $\mathcal{M}$ described above, we have $M = Term_{\mathcal{L}} = Var$ (notice that the equivalence relation defined above will be trivial in this case). Thus, since $(\mathcal{M}, s) \vDash \neg\mathsf{P}\mathsf{x}$ for all $\mathsf{x} \in Var$, it follows that $(\mathcal{M}, s) \nvDash \exists\mathsf{x}\mathsf{P}\mathsf{x}$. Hence, $\mathcal{M}$ is not a model of Δ.

The problem in the above example is that there was an existential statement in Δ, but whenever we plugged a term in for the quantified variable, the resulting formula was not in Δ. Since we are building our structure directly from the terms, this is a serious problem. However, if Δ had the following property, then this problem would not arise.

Definition 6.2.2. *Let $\mathcal{L}$ be a language and let $\Gamma \subseteq Form_{\mathcal{L}}$. We say that* Γ contains witnesses *if for all $\varphi \in Form_{\mathcal{L}}$ and all $\mathsf{x} \in Var$, there exists $\mathsf{c} \in \mathcal{C}$ such that $(\exists\mathsf{x}\varphi) \to \varphi^{\mathsf{c}}_{\mathsf{x}} \in \Gamma$.*

Our goal then is to show that if Γ is consistent, then there exists a $\Delta \supseteq \Gamma$ which is consistent, complete, and contains witnesses. On the face of it, this is not true, as the above example shows (because there are no constant symbols). However, if we allow ourselves to expand our language with new constant symbols, we can repeatedly add witnessing statements by using these fresh constant symbols as our witnesses. The key question we need to consider is the following. Suppose that $\mathcal{L}$ is a language and $\Gamma \subseteq Form_{\mathcal{L}}$ is consistent. If we expand the language $\mathcal{L}$ to a language $\mathcal{L}'$ obtained by adding a new constant symbol, is the set Γ still consistent when viewed as a set of $\mathcal{L}'$ formulas? It might seem absolutely harmless to add a new constant symbol about which we say nothing (and it's not hard very hard to see that it is *semantically* harmless), but we are introducing new deductions in $\mathcal{L}'$. We need a way to convert a possibly bad $\mathcal{L}'$-deduction into a similarly bad $\mathcal{L}$-deduction to argue that Γ is still consistent as a set of $\mathcal{L}'$-formulas.

We can also define substitution of variables for constants in the obvious recursive fashion.

Definition 6.2.3. *Let $\mathsf{z} \in Var$ and let $\mathsf{c} \in \mathcal{C}$ be a constant symbol. We define a function $Subst^{\mathsf{z}}_{\mathsf{c}}\colon Term_{\mathcal{L}} \to Term_{\mathcal{L}}$, where we use $t^{\mathsf{z}}_{\mathsf{c}}$ to denote $Subst^{\mathsf{z}}_{\mathsf{c}}(t)$, as follows:*

1. $\mathsf{d}^{\mathsf{z}}_{\mathsf{c}} = \begin{cases} \mathsf{z} & \text{if } \mathsf{d} = \mathsf{c} \\ \mathsf{d} & \text{otherwise} \end{cases}$
   for all $\mathsf{d} \in \mathcal{C}$.

2. $\mathsf{x}^{\mathsf{z}}_{\mathsf{c}} = \mathsf{x}$ *for all* $\mathsf{x} \in Var$.

3. $(\mathsf{f}t_1t_2\ldots t_k)^{\mathsf{z}}_{\mathsf{c}} = \mathsf{f}(t_1)^{\mathsf{z}}_{\mathsf{c}}(t_2)^{\mathsf{z}}_{\mathsf{c}}\cdots(t_k)^{\mathsf{z}}_{\mathsf{c}}$ *for all* $\mathsf{f} \in \mathcal{F}_k$ *and all* $t_1, t_2, \ldots, t_k \in Term_{\mathcal{L}}$.

We now extend our function to $Subst^{\mathsf{z}}_{\mathsf{c}}\colon Form_{\mathcal{L}} \to Form_{\mathcal{L}}$, *again denoted* $\varphi^{\mathsf{z}}_{\mathsf{c}}$, *as follows:*

1. $(\mathsf{R}u_1u_2\cdots u_k)^{\mathsf{z}}_{\mathsf{c}} = \mathsf{R}(u_1)^{\mathsf{z}}_{\mathsf{c}}(u_2)^{\mathsf{z}}_{\mathsf{c}}\cdots(u_k)^{\mathsf{z}}_{\mathsf{c}}$ *for all* $\mathsf{R} \in \mathcal{R}_k$ *and all* $u_1, u_2, \ldots, u_k \in Term_{\mathcal{L}}$.

2. *We define* $(= u_1u_2)^{\mathsf{z}}_{\mathsf{c}}$ *to be* $= (u_1)^{\mathsf{z}}_{\mathsf{c}}(u_2)^{\mathsf{z}}_{\mathsf{c}}$ *for all* $u_1, u_2 \in Term_{\mathcal{L}}$.

3. $(\neg\varphi)^{\mathsf{z}}_{\mathsf{c}} = \neg(\varphi^{\mathsf{z}}_{\mathsf{c}})$ *for all* $\varphi \in Form_{\mathcal{L}}$.

4. $(\diamond\varphi\psi)^{\mathsf{z}}_{\mathsf{c}} = \diamond\varphi^{\mathsf{z}}_{\mathsf{c}}\psi^{\mathsf{z}}_{\mathsf{c}}$ *for all* $\varphi, \psi \in Form_{\mathcal{L}}$ *and all* $\diamond \in \{\wedge, \vee, \to\}$.

5. $(\mathsf{Q}\mathsf{x}\varphi)^{\mathsf{z}}_{\mathsf{c}} = \mathsf{Q}\mathsf{x}(\varphi^{\mathsf{z}}_{\mathsf{c}})$.

Lemma 6.2.4. *Let* $\varphi \in Form_{\mathcal{L}}$, *let* $t \in Term_{\mathcal{L}}$, *let* $\mathsf{c} \in \mathcal{C}$, *and let* $\mathsf{x}, \mathsf{z} \in Var$. *Suppose that* $\mathsf{z} \notin OccurVar(\varphi)$. *We have the following:*

1. $(\varphi^t_{\mathsf{x}})^{\mathsf{z}}_{\mathsf{c}}$ *equals* $(\varphi^{\mathsf{z}}_{\mathsf{c}})^{t^{\mathsf{z}}_{\mathsf{c}}}_{\mathsf{x}}$.

2. *If* $ValidSubst^t_{\mathsf{x}}(\varphi) = 1$, *then* $ValidSubst^{t^{\mathsf{z}}_{\mathsf{c}}}_{\mathsf{x}}(\varphi^{\mathsf{z}}_{\mathsf{c}}) = 1$.

Proof. A straightforward induction. □

Lemma 6.2.5. *Let* $\mathcal{L}$ *be a language, and let* $\mathcal{L}'$ *be* $\mathcal{L}$ *together with a new constant symbol* c. *Suppose that*

$$\begin{gathered} S_0 \vdash_{\mathcal{L}'} \varphi_0 \\ S_1 \vdash_{\mathcal{L}'} \varphi_1 \\ S_2 \vdash_{\mathcal{L}'} \varphi_2 \\ \vdots \\ S_n \vdash_{\mathcal{L}'} \varphi_n \end{gathered}$$

is an $\mathcal{L}'$*-deduction. For any* $\mathsf{z} \in Var$ *with* $\mathsf{z} \notin \bigcup_{i=0}^{n} OccurVar(S_i, \varphi_i)$, *we have that*

$$\begin{gathered} (S_0)^{\mathsf{z}}_{\mathsf{c}} \vdash_{\mathcal{L}} (\varphi_0)^{\mathsf{z}}_{\mathsf{c}} \\ (S_1)^{\mathsf{z}}_{\mathsf{c}} \vdash_{\mathcal{L}} (\varphi_1)^{\mathsf{z}}_{\mathsf{c}} \\ (S_2)^{\mathsf{z}}_{\mathsf{c}} \vdash_{\mathcal{L}} (\varphi_2)^{\mathsf{z}}_{\mathsf{c}} \\ \vdots \\ (S_n)^{\mathsf{z}}_{\mathsf{c}} \vdash_{\mathcal{L}} (\varphi_n)^{\mathsf{z}}_{\mathsf{c}} \end{gathered}$$

is an $\mathcal{L}$*-deduction.*

Proof. We prove by induction on i that

$$\begin{gathered}(S_0)^{\mathsf{z}}_{\mathsf{c}} \vdash_{\mathcal{L}} (\varphi_0)^{\mathsf{z}}_{\mathsf{c}} \\ (S_1)^{\mathsf{z}}_{\mathsf{c}} \vdash_{\mathcal{L}} (\varphi_1)^{\mathsf{z}}_{\mathsf{c}} \\ (S_2)^{\mathsf{z}}_{\mathsf{c}} \vdash_{\mathcal{L}} (\varphi_2)^{\mathsf{z}}_{\mathsf{c}} \\ \vdots \\ (S_i)^{\mathsf{z}}_{\mathsf{c}} \vdash_{\mathcal{L}} (\varphi_i)^{\mathsf{z}}_{\mathsf{c}}\end{gathered}$$

is an $\mathcal{L}$-deduction.

If $\varphi \in S$, then $\varphi^{\mathsf{z}}_{\mathsf{c}} \in S^{\mathsf{z}}_{\mathsf{c}}$.

Suppose that line i is $S \vdash t = t$ where $t \in Term_{\mathcal{L}'}$. Since $(t = t)^{\mathsf{z}}_{\mathsf{c}}$ equals $t^{\mathsf{z}}_{\mathsf{c}} = t^{\mathsf{z}}_{\mathsf{c}}$, we can place $S^{\mathsf{z}}_{\mathsf{c}} \vdash t^{\mathsf{z}}_{\mathsf{c}} = t^{\mathsf{z}}_{\mathsf{c}}$ on line i by the $= Refl$ rule.

Suppose that $S \vdash_{\mathcal{L}'} \varphi \wedge \psi$ was a previous line and we inferred $S \vdash_{\mathcal{L}'} \varphi$. Inductively, we have $S^{\mathsf{z}}_{\mathsf{c}} \vdash_{\mathcal{L}} (\varphi \wedge \psi)^{\mathsf{z}}_{\mathsf{c}}$ on the corresponding line. Since $(\varphi \wedge \psi)^{\mathsf{z}}_{\mathsf{c}} = \varphi^{\mathsf{z}}_{\mathsf{c}} \wedge \psi^{\mathsf{z}}_{\mathsf{c}}$, we may use the $\wedge EL$ rule to put $S^{\mathsf{z}}_{\mathsf{c}} \vdash_{\mathcal{L}} \varphi^{\mathsf{z}}_{\mathsf{c}}$ on the corresponding line. The other propositional rules are similarly uninteresting.

Suppose that $S \vdash_{\mathcal{L}'} \varphi^t_{\mathsf{x}}$ and $S \vdash_{\mathcal{L}'} t = u$ were previous lines, that $ValidSubst^t_{\mathsf{x}}(\varphi) = 1 = ValidSubst^u_{\mathsf{x}}(\varphi)$, and we inferred $S \vdash_{\mathcal{L}'} \varphi^u_{\mathsf{x}}$. Inductively, we have $S^{\mathsf{z}}_{c} \vdash_{\mathcal{L}} (\varphi^t_{\mathsf{x}})^{\mathsf{z}}_{\mathsf{c}}$ and $S^{\mathsf{z}}_{\mathsf{c}} \vdash_{\mathcal{L}} (t = u)^{\mathsf{z}}_{\mathsf{c}}$ on the corresponding lines. Now $(\varphi^t_{\mathsf{x}})^{\mathsf{z}}_{\mathsf{c}}$ equals $(\varphi^{\mathsf{z}}_{\mathsf{c}})^{t^{\mathsf{z}}_{\mathsf{c}}}_{\mathsf{x}}$ by the previous lemma, and $(t = u)^{\mathsf{z}}_{\mathsf{c}}$ equals $t^{\mathsf{z}}_{\mathsf{c}} = u^{\mathsf{z}}_{\mathsf{c}}$. Thus, we have $S^{\mathsf{z}}_{c} \vdash_{\mathcal{L}} (\varphi^{\mathsf{z}}_{\mathsf{c}})^{t^{\mathsf{z}}_{\mathsf{c}}}_{\mathsf{x}}$ and $S^{\mathsf{z}}_{\mathsf{c}} \vdash_{\mathcal{L}} t^{\mathsf{z}}_{\mathsf{c}} = u^{\mathsf{z}}_{\mathsf{c}}$ on the corresponding lines. Using the fact that $ValidSubst^t_{\mathsf{x}}(\varphi) = 1 = ValidSubst^u_{\mathsf{x}}(\varphi)$, we can use the previous lemma to conclude that that $ValidSubst^{t^{\mathsf{z}}_{\mathsf{c}}}_{\mathsf{x}}(\varphi^{\mathsf{z}}_{\mathsf{c}}) = 1 = ValidSubst^{u^{\mathsf{z}}_{\mathsf{c}}}_{\mathsf{x}}(\varphi^{\mathsf{z}}_{\mathsf{c}})$. Hence, we may use that $= Sub$ rule to put $S^{\mathsf{z}}_{\mathsf{c}} \vdash_{\mathcal{L}} (\varphi^{\mathsf{z}}_{\mathsf{c}})^{u^{\mathsf{z}}_{\mathsf{c}}}_{\mathsf{x}}$ on the corresponding line. We now need only note that $(\varphi^{\mathsf{z}}_{\mathsf{c}})^{u^{\mathsf{z}}_{\mathsf{c}}}_{\mathsf{x}}$ equals $(\varphi^u_{\mathsf{x}})^{\mathsf{z}}_{\mathsf{c}}$ by the previous lemma.

Suppose that $S \vdash_{\mathcal{L}'} \varphi^t_{\mathsf{x}}$ where $ValidSubst^t_{\mathsf{x}}(\varphi) = 1$ was a previous line and we inferred $S \vdash_{\mathcal{L}'} \exists \mathsf{x} \varphi$. Inductively, we have $S^{\mathsf{z}}_{\mathsf{c}} \vdash_{\mathcal{L}} (\varphi^t_{\mathsf{x}})^{\mathsf{z}}_{\mathsf{c}}$ on the corresponding line. Now $(\varphi^t_{\mathsf{x}})^{\mathsf{z}}_{\mathsf{c}}$ equals $(\varphi^{\mathsf{z}}_{\mathsf{c}})^{t^{\mathsf{z}}_{\mathsf{c}}}_{\mathsf{x}}$ and $ValidSubst^{t^{\mathsf{z}}_{\mathsf{c}}}_{\mathsf{x}}(\varphi^{\mathsf{z}}_{\mathsf{c}}) = 1$ by the previous lemma. Hence, we may use the $\exists I$ rule to put $S^{\mathsf{z}}_{\mathsf{c}} \vdash_{\mathcal{L}} \exists \mathsf{x}(\varphi^{\mathsf{z}}_{\mathsf{c}})$ on the corresponding line. We now need only note that $\exists \mathsf{x}(\varphi^{\mathsf{z}}_{\mathsf{c}})$ equals $(\exists \mathsf{x} \varphi)^{\mathsf{z}}_{\mathsf{c}}$.

The other rules are similar. □

Corollary 6.2.6. *Let $\mathcal{L}$ be a language, let $\Gamma \subseteq Form_{\mathcal{L}}$, and let $\varphi \in Form_{\mathcal{L}}$.*

1. *Let $\mathcal{L}'$ be $\mathcal{L}$ together with a new constant symbol. If $\Gamma \vdash_{\mathcal{L}'} \varphi$, then $\Gamma \vdash_{\mathcal{L}} \varphi$.*
2. *Let $\mathcal{L}'$ be $\mathcal{L}$ together with finitely many new constant symbols. If $\Gamma \vdash_{\mathcal{L}'} \varphi$, then $\Gamma \vdash_{\mathcal{L}} \varphi$.*
3. *Let $\mathcal{L}'$ be $\mathcal{L}$ together with (perhaps infinitely many) new constant symbols. If $\Gamma \vdash_{\mathcal{L}'} \varphi$, then $\Gamma \vdash_{\mathcal{L}} \varphi$.*

Proof.

1. Since $\Gamma \vdash_{\mathcal{L}'} \varphi$, we may fix an $\mathcal{L}'$-deduction

$$\begin{gathered}S_0 \vdash_{\mathcal{L}'} \varphi_0 \\ S_1 \vdash_{\mathcal{L}'} \varphi_1 \\ S_2 \vdash_{\mathcal{L}'} \varphi_2 \\ \vdots \\ S_n \vdash_{\mathcal{L}'} \varphi_n\end{gathered}$$

such that each $S_i \subseteq Form^*_{\mathcal{L}'}$, and where $S_n \in \Gamma^*$ and $\varphi_n = \varphi$. Fix $\mathsf{y} \in Var$ such that

$$\mathsf{y} \notin \bigcup_{i=0}^{n} OccurVar(S_i, \varphi_i).$$

From Lemma 6.2.5, we have that

$$\begin{array}{c}(S_0)^{\mathsf{y}}_{\mathsf{c}} \vdash_{\mathcal{L}} (\varphi_0)^{\mathsf{y}}_{\mathsf{c}} \\ (S_1)^{\mathsf{y}}_{\mathsf{c}} \vdash_{\mathcal{L}} (\varphi_1)^{\mathsf{y}}_{\mathsf{c}} \\ (S_2)^{\mathsf{y}}_{\mathsf{c}} \vdash_{\mathcal{L}} (\varphi_2)^{\mathsf{y}}_{\mathsf{c}} \\ \vdots \\ (S_n)^{\mathsf{y}}_{\mathsf{c}} \vdash_{\mathcal{L}} (\varphi_n)^{\mathsf{y}}_{\mathsf{c}}\end{array}$$

is an $\mathcal{L}$-deduction. Since $S_n \in \Gamma^* \subseteq Form^*_{\mathcal{L}}$ and $\varphi \in Form_{\mathcal{L}}$, it follows that $(S_n)^{\mathsf{y}}_{\mathsf{c}} = S_n$ and $(\varphi_n)^{\mathsf{y}}_{\mathsf{c}} = \varphi$. Thus, $S_n \vdash_{\mathcal{L}} \varphi$ and so $\Gamma \vdash_{\mathcal{L}} \varphi$.

2. This is proved by induction on the number of new constant symbols, using part (1) for both the base case and the inductive step.

3. Since $\Gamma \vdash_{\mathcal{L}'} \varphi$, we may fix an $\mathcal{L}'$-deduction

$$\begin{array}{c}S_0 \vdash_{\mathcal{L}'} \varphi_0 \\ S_1 \vdash_{\mathcal{L}'} \varphi_1 \\ S_2 \vdash_{\mathcal{L}'} \varphi_2 \\ \vdots \\ S_n \vdash_{\mathcal{L}'} \varphi_n\end{array}$$

such that each $S_i \subseteq Form^*_{\mathcal{L}'}$, and where $S_n \in \Gamma^*$ and $\varphi_n = \varphi$. Let $\{\mathsf{c}_0, \mathsf{c}_1, \ldots, \mathsf{c}_m\}$ be all of the constant symbols appearing in some S_i or φ_i, and let $\mathcal{L}_0 = \mathcal{L} \cup \{\mathsf{c}_0, \mathsf{c}_1, \ldots, \mathsf{c}_m\}$. We then have that

$$\begin{array}{c}S_0 \vdash_{\mathcal{L}_0} \varphi_0 \\ S_1 \vdash_{\mathcal{L}_0} \varphi_1 \\ S_2 \vdash_{\mathcal{L}_0} \varphi_2 \\ \vdots \\ S_n \vdash_{\mathcal{L}_0} \varphi_n\end{array}$$

is an $\mathcal{L}_0$-deduction, so $\Gamma \vdash_{\mathcal{L}_0} \varphi$. Therefore, $\Gamma \vdash_{\mathcal{L}} \varphi$ by part (2).

□

Corollary 6.2.7. *Let $\mathcal{L}$ be a language and let $\mathcal{L}'$ be $\mathcal{L}$ together with (perhaps infinitely many) new constant symbols. Let $\Gamma \subseteq Form_{\mathcal{L}}$. Γ is $\mathcal{L}$-consistent if and only if Γ is $\mathcal{L}'$-consistent.*

Proof. Since any $\mathcal{L}$-deduction is also a $\mathcal{L}'$-deduction, if Γ is $\mathcal{L}$-inconsistent then it is trivially $\mathcal{L}'$-inconsistent. Suppose that Γ is $\mathcal{L}'$-inconsistent. We then have that $\Gamma \vdash_{\mathcal{L}'} \varphi$ for all $\varphi \in Form_{\mathcal{L}}$ by Proposition 6.1.14, hence $\Gamma \vdash_{\mathcal{L}} \varphi$ for all $\varphi \in Form_{\mathcal{L}}$ by Corollary 6.2.6. Therefore, Γ is $\mathcal{L}$-inconsistent. □

Corollary 6.2.8 (Generalization on Constants)**.** *Let $\mathcal{L}$ be a language, and let $\mathcal{L}'$ be $\mathcal{L}$ together with a new constant symbol c. Suppose that $\Gamma \subseteq Form_{\mathcal{L}}$ and $\varphi \in Form_{\mathcal{L}}$. If $\Gamma \vdash_{\mathcal{L}'} \varphi^{\mathsf{c}}_{\mathsf{x}}$, then $\Gamma \vdash_{\mathcal{L}} \forall \mathsf{x} \varphi$.*

Proof. Since $\Gamma \vdash_{\mathcal{L}'} \varphi^{\mathsf{c}}_{\mathsf{x}}$, we may fix an $\mathcal{L}'$-deduction

$$\begin{array}{c} S_0 \vdash_{\mathcal{L}'} \varphi_0 \\ S_1 \vdash_{\mathcal{L}'} \varphi_1 \\ S_2 \vdash_{\mathcal{L}'} \varphi_2 \\ \vdots \\ S_n \vdash_{\mathcal{L}'} \varphi_n \end{array}$$

such that each $S_i \subseteq Form^*_{\mathcal{L}'}$, and that $S_n \in \Gamma^*$ and $\varphi_n = \varphi^{\mathsf{c}}_{\mathsf{x}}$. Fix $\mathsf{y} \in Var$ such that

$$\mathsf{y} \notin \bigcup_{i=0}^{n} OccurVar(S_i, \varphi_i).$$

From Lemma 6.2.5, we have that

$$\begin{array}{c} (S_0)^{\mathsf{y}}_{\mathsf{c}} \vdash_{\mathcal{L}} (\varphi_0)^{\mathsf{y}}_{\mathsf{c}} \\ (S_1)^{\mathsf{y}}_{\mathsf{c}} \vdash_{\mathcal{L}} (\varphi_1)^{\mathsf{y}}_{\mathsf{c}} \\ (S_2)^{\mathsf{y}}_{\mathsf{c}} \vdash_{\mathcal{L}} (\varphi_2)^{\mathsf{y}}_{\mathsf{c}} \\ \vdots \\ (S_n)^{\mathsf{y}}_{\mathsf{c}} \vdash_{\mathcal{L}} (\varphi_n)^{\mathsf{y}}_{\mathsf{c}} \end{array}$$

is an $\mathcal{L}$-deduction. Since $S_n \in \Gamma^* \subseteq Form^*_{\mathcal{L}}$, we have $(S_n)^{\mathsf{y}}_{\mathsf{c}} = S_n$. Now $(\varphi_n)^{\mathsf{y}}_{\mathsf{c}} = (\varphi^{\mathsf{c}}_{\mathsf{x}})^{\mathsf{y}}_{\mathsf{c}} = \varphi^{\mathsf{y}}_{\mathsf{x}}$. We therefore have $S_n \vdash_{\mathcal{L}} \varphi^{\mathsf{y}}_{\mathsf{x}}$. We may then use the $\forall I$ rule to conclude that $S_n \vdash_{\mathcal{L}} \forall \mathsf{x}\varphi$. Since $S_n \in \Gamma^*$, it follows that $\Gamma \vdash_{\mathcal{L}} \forall \mathsf{x}\varphi$. □

Lemma 6.2.9. *For all $\varphi, \psi \in Form_{\mathcal{L}}$, we have $\neg(\varphi \to \psi) \vdash \varphi \wedge \neg\psi$.*

Proof. Let $\varphi, \psi \in Form_{\mathcal{L}}$ be arbitrary. Consider the following deduction:

$\neg(\varphi \to \psi), \neg\varphi, \varphi, \neg\psi \vdash \varphi$	$(Assume_{\mathcal{L}})$	(1)
$\neg(\varphi \to \psi), \neg\varphi, \varphi, \neg\psi \vdash \neg\varphi$	$(Assume_{\mathcal{L}})$	(2)
$\neg(\varphi \to \psi), \neg\varphi, \varphi \vdash \psi$	($Contr$ on 1 and 2)	(3)
$\neg(\varphi \to \psi), \neg\varphi \vdash \varphi \to \psi$	($\to I$ on 3)	(4)
$\neg(\varphi \to \psi), \neg\varphi \vdash \neg(\varphi \to \psi)$	$(Assume_{\mathcal{L}})$	(5)
$\neg(\varphi \to \psi) \vdash \varphi$	($Contr$ on 4 and 5)	(6)
$\neg(\varphi \to \psi), \neg\neg\psi, \varphi, \neg\psi \vdash \neg\psi$	$(Assume_{\mathcal{L}})$	(7)
$\neg(\varphi \to \psi), \neg\neg\psi, \varphi, \neg\psi \vdash \neg\neg\psi$	$(Assume_{\mathcal{L}})$	(8)
$\neg(\varphi \to \psi), \neg\neg\psi, \varphi \vdash \psi$	($Contr$ on 7 and 8)	(9)
$\neg(\varphi \to \psi), \neg\neg\psi \vdash \varphi \to \psi$	($\to I$ on 9)	(10)
$\neg(\varphi \to \psi), \neg\neg\psi \vdash \neg(\varphi \to \psi)$	$(Assume_{\mathcal{L}})$	(11)
$\neg(\varphi \to \psi) \vdash \neg\psi$	($Contr$ on 10 and 11)	(12)
$\neg(\varphi \to \psi) \vdash \varphi \wedge \neg\psi$	($\wedge I$ on 6 and 12)	(13)

□

Lemma 6.2.10. *Let $\mathcal{L}$ be a language, and let $\mathcal{L}'$ be $\mathcal{L}$ together with a new constant symbol c. Let $\Gamma \subseteq Form_{\mathcal{L}}$ and let $\varphi \in Form_{\mathcal{L}}$. If Γ is $\mathcal{L}$-consistent, then $\Gamma \cup \{(\exists \mathsf{x}\varphi) \to \varphi^{\mathsf{c}}_{\mathsf{x}}\}$ is $\mathcal{L}'$-consistent.*

Proof. We prove the contrapositive. Suppose then that $\Gamma \cup \{(\exists \mathsf{x}\varphi) \to \varphi^{\mathsf{c}}_{\mathsf{x}}\}$ is $\mathcal{L}'$-inconsistent. By Proposition 6.1.15, we then have that

$$\Gamma \vdash_{\mathcal{L}'} \neg((\exists \mathsf{x}\varphi) \to \varphi^{\mathsf{c}}_{\mathsf{x}}).$$

From Lemma 6.2.9, we have

$$\neg((\exists x\varphi) \to \varphi^{c}_{x}) \vdash_{\mathcal{L}'} (\exists x\varphi) \wedge \neg(\varphi^{c}_{x}),$$

and hence

$$\Gamma \cup \{\neg((\exists x\varphi) \to \varphi^{c}_{x})\} \vdash_{\mathcal{L}'} (\exists x\varphi) \wedge \neg(\varphi^{c}_{x}).$$

Using Proposition 6.1.17, we conclude that

$$\Gamma \vdash_{\mathcal{L}'} (\exists x\varphi) \wedge \neg(\varphi^{c}_{x}).$$

Using the $\wedge EL$ rule, we conclude that $\Gamma \vdash_{\mathcal{L}'} \exists \mathsf{x}\varphi$. Since $\exists \mathsf{x}\varphi \vdash_{\mathcal{L}'} \neg\forall \mathsf{x}\neg\varphi$ by Proposition 6.1.9, we can again use Proposition 6.1.17 to conclude that $\Gamma \vdash_{\mathcal{L}'} \neg\forall \mathsf{x}\neg\varphi$. Since $\Gamma \subseteq Form_{\mathcal{L}}$ and $\neg\forall \mathsf{x}\neg\varphi \in Form_{\mathcal{L}}$, Corollary 6.2.6 allows us to conclude that

$$\Gamma \vdash_{\mathcal{L}} \neg\forall \mathsf{x}\neg\varphi.$$

Using the $\wedge ER$ rule instead, together with the fact that $\neg(\varphi^{\mathsf{c}}_{\mathsf{x}})$ equals $(\neg\varphi)^{\mathsf{c}}_{\mathsf{x}}$, we also have $\Gamma \vdash_{\mathcal{L}'} (\neg\varphi)^{\mathsf{c}}_{\mathsf{x}}$, so

$$\Gamma \vdash_{\mathcal{L}} \forall \mathsf{x}\neg\varphi$$

by Generalization on Constants. Therefore, Γ is $\mathcal{L}$-inconsistent. □

Lemma 6.2.11. *Let $\mathcal{L}$ be a language and let $\Gamma \subseteq Form_{\mathcal{L}}$ be $\mathcal{L}$-consistent. There exists a language $\mathcal{L}' \supseteq \mathcal{L}$, obtained from $\mathcal{L}$ by only adding constant symbols, and $\Gamma' \subseteq Form_{\mathcal{L}'}$ with the following properties:*

1. *$\Gamma \subseteq \Gamma'$.*
2. *Γ' is $\mathcal{L}'$-consistent.*
3. *For all $\varphi \in Form_{\mathcal{L}}$ and all $\mathsf{x} \in Var$, there exists $\mathsf{c} \in \mathcal{C}$ such that $(\exists \mathsf{x}\varphi) \to \varphi^{\mathsf{c}}_{\mathsf{x}} \in \Gamma'$.*

Proof. For each $\varphi \in Form_{\mathcal{L}}$ and each $\mathsf{x} \in Var$, let $\mathsf{c}_{\varphi,\mathsf{x}}$ be a new constant symbol (distinct from all symbols in $\mathcal{L}$). Let $\mathcal{L}' = \mathcal{L} \cup \{\mathsf{c}_{\varphi,\mathsf{x}} : \varphi \in Form_{\mathcal{L}} \text{ and } \mathsf{x} \in Var\}$. Let

$$\Gamma' = \Gamma \cup \{(\exists \mathsf{x}\varphi) \to \varphi^{\mathsf{c}_{\varphi,\mathsf{x}}}_{\mathsf{x}} : \varphi \in Form_{\mathcal{L}} \text{ and } \mathsf{x} \in Var\}.$$

Conditions 1 and 3 are clear, so we need only check that Γ' is $\mathcal{L}'$-consistent. By Corollary 6.1.19, it suffices to check that all finite subsets of Γ' are $\mathcal{L}'$-consistent, and for this it suffices to show that

$$\Gamma \cup \{(\exists \mathsf{x}_1\varphi_1) \to (\varphi_1)^{\mathsf{c}_{\varphi_1,\mathsf{x}_1}}_{\mathsf{x}_1}, (\exists \mathsf{x}_2\varphi_2) \to (\varphi_2)^{\mathsf{c}_{\varphi_2,\mathsf{x}_2}}_{\mathsf{x}_2}, \ldots, (\exists \mathsf{x}_\mathsf{n}\varphi_\mathsf{n}) \to (\varphi_\mathsf{n})^{\mathsf{c}_{\varphi_\mathsf{n},\mathsf{x}_\mathsf{n}}}_{\mathsf{x}_\mathsf{n}}\}$$

is $\mathcal{L}'$-consistent whenever $\varphi_1, \varphi_2, \ldots, \varphi_n \in Form_{\mathcal{L}}$ and $\mathsf{x}_1, \mathsf{x}_2, \ldots, \mathsf{x}_n \in Var$. Formally, one can prove this by induction on n. A slightly informal argument is as as follows. Let $\varphi_1, \varphi_2, \ldots, \varphi_n \in Form_{\mathcal{L}}$ and $\mathsf{x}_1, \mathsf{x}_2, \ldots, \mathsf{x}_\mathsf{n} \in Var$ be arbitrary. Since Γ is $\mathcal{L}$-consistent, we can apply Lemma 6.2.10 to conclude that

$$\Gamma \cup \{(\exists \mathsf{x}_1\varphi_1) \to (\varphi_1)^{\mathsf{c}_{\varphi_1,\mathsf{x}_1}}_{\mathsf{x}_1}\}$$

is $(\mathcal{L} \cup \{\mathsf{c}_{\varphi_1,\mathsf{x}_1}\})$-consistent. Applying Lemma 6.2.10, we conclude that

$$\Gamma \cup \{(\exists \mathsf{x}_1\varphi_1) \to (\varphi_1)^{\mathsf{c}_{\varphi_1,\mathsf{x}_1}}_{\mathsf{x}_1}, (\exists \mathsf{x}_2\varphi_2) \to (\varphi_2)^{\mathsf{c}_{\varphi_2,\mathsf{x}_2}}_{\mathsf{x}_2}\}$$

is $(\mathcal{L} \cup \{\mathsf{c}_{\varphi_1,\mathsf{x}_1}, \mathsf{c}_{\varphi_2,\mathsf{x}_2}\})$. By repeated applications of Lemma 6.2.10, we eventually conclude that

$$\Gamma \cup \{(\exists \mathsf{x}_1\varphi_1) \to (\varphi_1)^{\mathsf{c}_{\varphi_1,\mathsf{x}_1}}_{\mathsf{x}_1}, (\exists \mathsf{x}_2\varphi_2) \to (\varphi_2)^{\mathsf{c}_{\varphi_2,\mathsf{x}_2}}_{\mathsf{x}_2}, \ldots, (\exists \mathsf{x}_\mathsf{n}\varphi_\mathsf{n}) \to (\varphi_\mathsf{n})^{\mathsf{c}_{\varphi_\mathsf{n},\mathsf{x}_\mathsf{n}}}_{\mathsf{x}_\mathsf{n}}\}$$

is $(\mathcal{L} \cup \{\mathsf{c}_{\varphi_1,\mathsf{x}_1}, \mathsf{c}_{\varphi_2,\mathsf{x}_2}, \ldots, \mathsf{c}_{\varphi_\mathsf{n},\mathsf{x}_\mathsf{n}}\})$-consistent. Therefore,

$$\Gamma \cup \{(\exists \mathsf{x}_1\varphi_1) \to (\varphi_1)^{\mathsf{c}_{\varphi_1,\mathsf{x}_1}}_{\mathsf{x}_1}, (\exists \mathsf{x}_2\varphi_2) \to (\varphi_2)^{\mathsf{c}_{\varphi_2,\mathsf{x}_2}}_{\mathsf{x}_2}, \ldots, (\exists \mathsf{x}_\mathsf{n}\varphi_\mathsf{n}) \to (\varphi_\mathsf{n})^{\mathsf{c}_{\varphi_\mathsf{n},\mathsf{x}_\mathsf{n}}}_{\mathsf{x}_\mathsf{n}}\}$$

is $\mathcal{L}'$-consistent by Corollary 6.2.7, which completes the proof by the above comments. □

Proposition 6.2.12. *Let $\mathcal{L}$ be a language and let $\Gamma \subseteq Form_{\mathcal{L}}$ be consistent. There exists a language $\mathcal{L}' \supseteq \mathcal{L}$, obtained from $\mathcal{L}$ by only adding constant symbols, and $\Gamma' \subseteq Form_{\mathcal{L}'}$ with the following properties:*

1. $\Gamma \subseteq \Gamma'$.
2. Γ' *is* $\mathcal{L}'$*-consistent.*
3. Γ' *contains witnesses.*

Proof. Let $\mathcal{L}_0 = \mathcal{L}$ and $\Gamma_0 = \Gamma$. For each $n \in \mathbb{N}$, use Lemma 6.2.11 to obtain $\mathcal{L}_{n+1}$ and Γ_{n+1} from $\mathcal{L}_n$ and Γ_n. Now let $\mathcal{L}' = \bigcup_{n\in\mathbb{N}} \mathcal{L}$ and set $\Gamma' = \bigcup_{n\in\mathbb{N}} \Gamma_n$. We then clearly have properties (1) and (3), and property (2) follows from Corollary 6.1.19 and Corollary 6.2.7. □

Proposition 6.2.13. *If Γ is consistent, then there exists a set $\Delta \supseteq \Gamma$ which is consistent and complete.*

Proof. Exactly the same proof as the propositional logic case, using Zorn's Lemma in the uncountable case (see Proposition 3.5.4 and Proposition 3.5.7). □

Proposition 6.2.14. *Let $\mathcal{L}$ be a language. If $\Gamma \subseteq \mathcal{L}$ is consistent, then there a language $\mathcal{L}' \supseteq \mathcal{L}$, obtained from $\mathcal{L}$ by only adding constant symbols, and $\Delta \subseteq Form_{\mathcal{L}'}$ with the following properties:*

- $\Gamma \subseteq \Delta$.
- Δ *is consistent.*
- Δ *is complete.*
- Δ *contains witnesses.*

Proof. First apply Proposition 6.2.12 to obtain $\mathcal{L}'$ and Γ', and then apply Proposition 6.2.13 to obtain Δ from Γ' □

Lemma 6.2.15. *Suppose that Δ is consistent and complete. If $\Delta \vdash \varphi$, then $\varphi \in \Delta$.*

Proof. Suppose that $\Delta \vdash \varphi$. Since Δ is complete, we have that either $\varphi \in \Delta$ or $\neg\varphi \in \Delta$. Now if $\neg\varphi \in \Delta$, then we would would trivially have $\Delta \vdash \neg\varphi$ (in addition to our assumed $\Delta \vdash \varphi$), contradicting the fact that Δ is consistent. It follows that $\varphi \in \Delta$. □

Lemma 6.2.16. *Suppose that Δ is consistent, complete, and contains witnesses. For every $t \in Term_{\mathcal{L}}$, there exists $\mathsf{c} \in \mathcal{C}$ such that $t = \mathsf{c} \in \Delta$.*

Proof. Let $t \in Term_{\mathcal{L}}$. Fix $\mathsf{x} \in Var$ such that $\mathsf{x} \notin OccurVar(t)$. Since Δ contains witnesses, we may fix $\mathsf{c} \in \mathcal{C}$ such that $(\exists \mathsf{x}(t = \mathsf{x})) \to (t = \mathsf{c}) \in \Delta$ (using the formula $t = \mathsf{x}$). Now $\Delta \vdash (t = \mathsf{x})^t_{\mathsf{x}}$, so we may use the $\exists I$ rule (because $ValidSubst^t_{\mathsf{x}}(t = \mathsf{x}) = 1$) to conclude that $\Delta \vdash \exists \mathsf{x}(t = \mathsf{x})$. Since we have both $\Delta \vdash \exists \mathsf{x}(t = \mathsf{x})$ and $\Delta \vdash (\exists \mathsf{x}(t = \mathsf{x})) \to (t = \mathsf{c})$, we can apply Proposition 6.1.17 to conclude that $\Delta \vdash t = \mathsf{c}$. Using Lemma 6.2.15, it follows that that $t = \mathsf{c} \in \Delta$. □

Lemma 6.2.17. *Suppose that Δ is consistent, complete, and contains witnesses. We have*

1. $\neg\varphi \in \Delta$ *if and only if* $\varphi \notin \Delta$.
2. $\varphi \wedge \psi \in \Delta$ *if and only if* $\varphi \in \Delta$ *and* $\psi \in \Delta$.
3. $\varphi \vee \psi \in \Delta$ *if and only if* $\varphi \in \Delta$ *or* $\psi \in \Delta$.
4. $\varphi \to \psi \in \Delta$ *if and only if* $\varphi \notin \Delta$ *or* $\psi \in \Delta$.

5. $\exists \mathsf{x}\varphi \in \Delta$ *if and only if there exists* $\mathsf{c} \in \mathcal{C}$ *such that* $\varphi^{\mathsf{c}}_{\mathsf{x}} \in \Delta$.

6. $\forall \mathsf{x}\varphi \in \Delta$ *if and only if* $\varphi^{\mathsf{c}}_{\mathsf{x}} \in \Delta$ *for all* $\mathsf{c} \in \mathcal{C}$.

Proof. The proofs of the first four statements are identical to the proofs in the propositional logic case (see Lemma 3.5.9). We prove the last two.

5. Suppose first that $\exists \mathsf{x}\varphi \in \Delta$. Since Δ contains witnesses, we may fix $\mathsf{c} \in \mathcal{C}$ such that $(\exists \mathsf{x}\varphi) \to \varphi^{\mathsf{c}}_{\mathsf{x}} \in \Delta$. We therefore have $\Delta \vdash \exists \mathsf{x}\varphi$ and $\Delta \vdash (\exists \mathsf{x}\varphi) \to \varphi^{\mathsf{c}}_{\mathsf{x}}$, hence $\Delta \vdash \varphi^{\mathsf{c}}_{\mathsf{x}}$ by Proposition 6.1.17. Using Lemma 6.2.15, we conclude that $\varphi^{\mathsf{c}}_{\mathsf{x}} \in \Delta$.

   Conversely, suppose that there exists $\mathsf{c} \in \mathcal{C}$ such that $\varphi^{\mathsf{c}}_{\mathsf{x}} \in \Delta$. We then have $\Delta \vdash \varphi^{\mathsf{c}}_{\mathsf{x}}$, hence $\Delta \vdash \exists \mathsf{x}\varphi$ using the $\exists I$ rule (notice that $ValidSubst^{\mathsf{c}}_{\mathsf{x}}(\varphi) = 1$). Using Lemma 6.2.15, we conclude that $\exists \mathsf{x}\varphi \in \Delta$.

6. Suppose first that $\forall \mathsf{x}\varphi \in \Delta$. We then have $\Delta \vdash \forall \mathsf{x}\varphi$, hence $\Delta \vdash \varphi^{\mathsf{c}}_{\mathsf{x}}$ for all $\mathsf{c} \in \mathcal{C}$ using the $\forall E$ rule (notice that $ValidSubst^{\mathsf{c}}_{\mathsf{x}}(\varphi) = 1$ for all $\mathsf{c} \in \mathcal{C}$). Using Lemma 6.2.15, we conclude that $\varphi^{\mathsf{c}}_{\mathsf{x}} \in \Delta$ for all $c \in \mathcal{C}$.

   Conversely, suppose that $\varphi^{\mathsf{c}}_{\mathsf{x}} \in \Delta$ for all $\mathsf{c} \in \mathcal{C}$. Since Δ is consistent, this implies that there does not exist $\mathsf{c} \in \mathcal{C}$ with $\neg(\varphi^{\mathsf{c}}_{\mathsf{x}}) = (\neg\varphi)^{\mathsf{c}}_{\mathsf{x}} \in \Delta$. Therefore, $\exists \mathsf{x}\neg\varphi \notin \Delta$ by part 5, so $\neg\exists \mathsf{x}\neg\varphi \in \Delta$ by part (1). It follows from Proposition 6.1.10 that $\Delta \vdash \forall \mathsf{x}\varphi$. Using Lemma 6.2.15, we conclude that $\forall \mathsf{x}\varphi \in \Delta$.

□

Proposition 6.2.18. *If* Δ *is consistent, complete, and contains witnesses, then* Δ *is satisfiable.*

Proof. Suppose that Δ is consistent, complete, and contains witnesses. Define a relation $\sim$ on $Term_{\mathcal{L}}$ by letting $t \sim u$ if $t = u \in \Delta$. We first check that $\sim$ is an equivalence relation. Reflexivity follows from the $= Refl$ rule and Lemma 6.2.15. Symmetry and transitivity follow from Proposition 6.1.5 and Proposition 6.1.6, together with Lemma 6.2.15.

We now define our $\mathcal{L}$-structure $\mathcal{M}$. We first let $M = Term_{\mathcal{L}}/\sim$ be the set of all equivalence classes of our equivalence relation. For each $t \in Term_{\mathcal{L}}$, we let $[t]$ denote the equivalence class of t. Notice that $M = \{[\mathsf{c}] : \mathsf{c} \in \mathcal{C}\}$ by Lemma 6.2.16. We now finish our description of the $\mathcal{L}$-structure $\mathcal{M}$ by saying how to interpret the constant, relation, and function symbols. We define the following:

1. $\mathsf{c}^{\mathcal{M}} = [\mathsf{c}]$ for all $\mathsf{c} \in \mathcal{C}$.

2. $\mathsf{R}^{\mathcal{M}} = \{([t_1], [t_2], \ldots, [t_k]) \in M^k : \mathsf{R}t_1t_2\cdots t_k \in \Delta\}$ for all $\mathsf{R} \in \mathcal{R}_k$.

3. $\mathsf{f}^{\mathcal{M}}([t_1], [t_2], \ldots, [t_k]) = [\mathsf{f}t_1t_2\cdots t_k]$ for all $\mathsf{f} \in \mathcal{F}_k$.

Notice that our definitions of $\mathsf{R}^{\mathcal{M}}$ do not depend on our choice of representatives for the equivalence classes by Proposition 6.1.7. Similarly, our definitions of $\mathsf{f}^{\mathcal{M}}$ do not depend on our choice of representatives for the equivalences classes by Proposition 6.1.8. Finally, define $s\colon Var \to M$ by letting $s(\mathsf{x}) = [\mathsf{x}]$ for all $\mathsf{x} \in Var$.

We first show that $\overline{s}(t) = [t]$ for all $t \in Term_{\mathcal{L}}$ by induction. We have $\overline{s}(\mathsf{c}) = \mathsf{c}^{\mathcal{M}} = [\mathsf{c}]$ for all $\mathsf{c} \in \mathcal{C}$ and $\overline{s}(\mathsf{x}) = s(\mathsf{x}) = [\mathsf{x}]$ for all $\mathsf{x} \in Var$. Suppose that $\mathsf{f} \in \mathcal{F}_k$ and $t_1, t_2, \ldots, t_k \in Term_{\mathcal{L}}$ are such that $\overline{s}(t_i) = [t_i]$ for all i. We then have

$$\begin{aligned}
\overline{s}(\mathsf{f}t_1t_2\cdots t_k) &= \mathsf{f}^{\mathcal{M}}(\overline{s}(t_1), \overline{s}(t_2), \ldots, \overline{s}(t_k)) \\
&= f^{\mathcal{M}}([t_1], [t_2], \ldots, [t_k]) && \text{(by induction)} \\
&= [\mathsf{f}t_1t_2\cdots t_k]
\end{aligned}$$

Therefore, $\overline{s}(t) = [t]$ for all $t \in Term_{\mathcal{L}}$.

We now show that by induction that for all $\varphi \in Form_{\mathcal{L}}$, we have $\varphi \in \Delta$ if and only if $(\mathcal{M}, s) \vDash \varphi$. We first prove the result for $\varphi \in AtomicForm_{\mathcal{L}}$. Given arbitrary $\mathsf{R} \in \mathcal{R}_k$ and $t_1, t_2, \ldots, t_k \in Term_{\mathcal{L}}$, we have

$$\begin{aligned} \mathsf{R}t_1t_2\cdots t_k \in \Delta &\Leftrightarrow ([t_1], [t_2], \ldots, [t_k]) \in \mathsf{R}^{\mathcal{M}} \\ &\Leftrightarrow (\overline{s}(t_1), \overline{s}(t_2), \ldots, \overline{s}(t_k)) \in \mathsf{R}^{\mathcal{M}} \\ &\Leftrightarrow (\mathcal{M}, s) \vDash \mathsf{R}t_1t_2\cdots t_k. \end{aligned}$$

Given arbitrary $t_1, t_2 \in Term_{\mathcal{L}}$, we have

$$\begin{aligned} t_1 = t_2 \in \Delta &\Leftrightarrow [t_1] = [t_2] \\ &\Leftrightarrow \overline{s}(t_1) = \overline{s}(t_2) \\ &\Leftrightarrow (\mathcal{M}, s) \vDash t_1 = t_2. \end{aligned}$$

If the statement is true for φ, then

$$\begin{aligned} \neg\varphi \in \Delta &\Leftrightarrow \varphi \notin \Delta && \text{(by Lemma 6.2.17)} \\ &\Leftrightarrow (\mathcal{M}, s) \nvDash \varphi && \text{(by induction)} \\ &\Leftrightarrow (\mathcal{M}, s) \vDash \varphi. \end{aligned}$$

Similarly, if the statement is true for φ and ψ, then

$$\begin{aligned} \varphi \wedge \psi \in \Delta &\Leftrightarrow \varphi \in \Delta \text{ and } \psi \in \Delta && \text{(by Lemma 6.2.17)} \\ &\Leftrightarrow (\mathcal{M}, s) \vDash \varphi \text{ and } (\mathcal{M}, s) \vDash \psi && \text{(by induction)} \\ &\Leftrightarrow (\mathcal{M}, s) \vDash \varphi \wedge \psi \end{aligned}$$

and

$$\begin{aligned} \varphi \vee \psi \in \Delta &\Leftrightarrow \varphi \in \Delta \text{ or } \psi \in \Delta && \text{(by Lemma 6.2.17)} \\ &\Leftrightarrow (\mathcal{M}, s) \vDash \varphi \text{ or } (\mathcal{M}, s) \vDash \psi && \text{(by induction)} \\ &\Leftrightarrow (\mathcal{M}, s) \vDash \varphi \vee \psi \end{aligned}$$

and finally

$$\begin{aligned} \varphi \to \psi \in \Delta &\Leftrightarrow \varphi \notin \Delta \text{ or } \psi \in \Delta && \text{(by Lemma 6.2.17)} \\ &\Leftrightarrow (\mathcal{M}, s) \nvDash \varphi \text{ or } (\mathcal{M}, s) \vDash \psi && \text{(by induction)} \\ &\Leftrightarrow (\mathcal{M}, s) \vDash \varphi \to \psi. \end{aligned}$$

If the statement is true for φ and $\mathsf{x} \in Var$ is arbitrary, then

$$\begin{aligned} \exists\mathsf{x}\varphi \in \Delta &\Leftrightarrow \text{There exists } \mathsf{c} \in \mathcal{C} \text{ such that } \varphi^{\mathsf{c}}_{\mathsf{x}} \in \Delta && \text{(by Lemma 6.2.17)} \\ &\Leftrightarrow \text{There exists } \mathsf{c} \in \mathcal{C} \text{ such that } (\mathcal{M}, s) \vDash \varphi^{\mathsf{c}}_{\mathsf{x}} && \text{(by induction)} \\ &\Leftrightarrow \text{There exists } \mathsf{c} \in \mathcal{C} \text{ such that } (\mathcal{M}, s[\mathsf{x} \Rightarrow \overline{s}(\mathsf{c})]) \vDash \varphi && \text{(by the Substitution Theorem)} \\ &\Leftrightarrow \text{There exists } \mathsf{c} \in \mathcal{C} \text{ such that } (\mathcal{M}, s[\mathsf{x} \Rightarrow [\mathsf{c}]]) \vDash \varphi \\ &\Leftrightarrow \text{There exists } a \in M \text{ such that } (\mathcal{M}, s[\mathsf{x} \Rightarrow a]) \vDash \varphi && \text{(since } M = \{[\mathsf{c}] : \mathsf{c} \in \mathcal{C}\}) \\ &\Leftrightarrow (\mathcal{M}, s) \vDash \exists\mathsf{x}\varphi \end{aligned}$$

and also

$$\begin{aligned}
\forall x\varphi \in \Delta &\Leftrightarrow \text{For all } \mathsf{c} \in \mathcal{C}, \text{ we have } \varphi_{\mathsf{x}}^{\mathsf{c}} \in \Delta && \text{(by Lemma 6.2.17)} \\
&\Leftrightarrow \text{For all } \mathsf{c} \in \mathcal{C}, \text{ we have } (\mathcal{M}, s) \vDash \varphi_{\mathsf{x}}^{\mathsf{c}} && \text{(by induction)} \\
&\Leftrightarrow \text{For all } \mathsf{c} \in \mathcal{C}, \text{ we have } (\mathcal{M}, s[\mathsf{x} \Rightarrow \overline{s}(\mathsf{c})]) \vDash \varphi && \text{(by the Substitution Theorem)} \\
&\Leftrightarrow \text{For all } \mathsf{c} \in \mathcal{C}, \text{ we have } (\mathcal{M}, s[\mathsf{x} \Rightarrow [\mathsf{c}]]) \\
&\Leftrightarrow \text{For all } a \in M, \text{ we have } (\mathcal{M}, s[\mathsf{x} \Rightarrow a]) \vDash \varphi && \text{(since } M = \{[\mathsf{c}] : \mathsf{c} \in \mathcal{C}\}) \\
&\Leftrightarrow (\mathcal{M}, s) \vDash \forall \mathsf{x}\varphi.
\end{aligned}$$

Therefore, by induction, we have $\varphi \in \Delta$ if and only if $(\mathcal{M}, s) \vDash \varphi$. In particular, we have $(\mathcal{M}, s) \vDash \varphi$ for all $\varphi \in \Delta$, hence Δ is satisfiable. □

Theorem 6.2.19 (Completeness Theorem)**.** *Let $\mathcal{L}$ be a language.*

1. *Every consistent set of formulas is satisfiable.*
2. *If $\Gamma \vDash \varphi$, then $\Gamma \vdash \varphi$.*

Proof.

1. Suppose that Γ is consistent. By Proposition 6.2.14, we may fix a language $\mathcal{L}' \supseteq \mathcal{L}$ and $\Delta \subseteq Form_{\mathcal{L}'}$ such that $\Delta \supseteq \Gamma$ is consistent, complete, and contains witnesses. Now Δ is satisfiable by Proposition 6.2.18, so we may fix an $\mathcal{L}'$-structure $\mathcal{M}'$ together with $s\colon Var \to M'$ such that $(\mathcal{M}', s) \vDash \varphi$ for all $\varphi \in \Delta$. We then have $(\mathcal{M}', s) \vDash \gamma$ for all $\gamma \in \Gamma$. Letting $\mathcal{M}$ be the restriction of $\mathcal{M}'$ to $\mathcal{L}$, we then have $(\mathcal{M}, s) \vDash \gamma$ for all $\gamma \in \Gamma$. Therefore, Γ is satisfiable.
2. Suppose that $\Gamma \vDash \varphi$. We then have that $\Gamma \cup \{\neg\varphi\}$ is unsatisfiable, hence $\Gamma \cup \{\neg\varphi\}$ is inconsistent by part 1. It follows from Proposition 6.1.15 that $\Gamma \vdash \varphi$.

□

We now give another proof of the Countable Lowenheim-Skolem Theorem which does not go through the concept of elementary substructures.

Corollary 6.2.20 (Countable Lowenheim-Skolem Theorem)**.** *Suppose that $\mathcal{L}$ is countable and $\Gamma \subseteq Form_{\mathcal{L}}$ is consistent. There exists a countable model of Γ.*

Proof. Notice that if $\mathcal{L}$ is consistent, then the $\mathcal{L}'$ formed in Lemma 6.2.11 is countable because $Form_{\mathcal{L}} \times Var$ is countable. Thus, each $\mathcal{L}_n$ in the proof of Proposition 6.2.12 is countable, so the $\mathcal{L}'$ formed in Proposition 6.2.12 is countable. It follows that $Term_{\mathcal{L}'}$ is countable, and since the $\mathcal{L}'$-structure $\mathcal{M}$ we construct in the proof of Proposition 6.2.18 is formed by taking the quotient from an equivalence relation on the countable set $Term_{\mathcal{L}'}$, we can conclude that $\mathcal{M}$ is countable. Therefore, the $\mathcal{L}$-structure which is the restriction of $\mathcal{M}$ to $\mathcal{L}$ from the proof of the Completeness Theorem is countable. □

6.3 Compactness and Applications

Now that we have completed proofs of the Soundness and Completeness Theorems, we immediately obtain the following result, which is one of the primary tools in logic.

Corollary 6.3.1 (Compactness Theorem)**.** *Let $\mathcal{L}$ be a language.*

1. *If $\Gamma \vDash \varphi$, then there exists a finite $\Gamma_0 \subseteq \Gamma$ such that $\Gamma_0 \vDash \varphi$.*

2. *If every finite subset of* Γ *is satisfiable, then* Γ *is satisfiable.*

Proof.

1. Suppose that $\Gamma \vDash \varphi$. By the Completeness Theorem, we have $\Gamma \vdash \varphi$. Using Proposition 6.1.18, we may fix a finite $\Gamma_0 \subseteq \Gamma$ such that $\Gamma_0 \vdash \varphi$. By the Soundness Theorem, we have $\Gamma_0 \vDash \varphi$.

2. If every finite subset of Γ is satisfiable, then every finite subset of Γ is consistent by the Soundness Theorem, hence Γ is consistent by Corollary 6.1.19, and so Γ is satisfiable by the Soundness Theorem.

□

For our first application of Compactness, we prove another result expressing the fact that first-order logic is not powerful enough to distinguish certain aspects of cardinality. We already have the Lowenheim-Skolem Theorem saying that any satisfiable set has a countable model (assuming that $\mathcal{L}$ is countable), and hence first-order logic does not have the expressive power to force all models to be uncountable. In this case, our distinction is between large finite numbers and the infinite.

Proposition 6.3.2. *Let* $\mathcal{L}$ *be a language. Suppose that* $\Gamma \subseteq Form_{\mathcal{L}}$ *is such that for all* $n \in \mathbb{N}$*, there exists a model* $(\mathcal{M}, s)$ *of* Γ *such that* $|M| > n$*. We then that there exists a model* $(\mathcal{M}, s)$ *of* Γ *such that* M *is infinite.*

We give two proofs.

Proof 1. For each $n \in \mathbb{N}$ with $n \geq 2$, let σ_n be the sentence

$$\exists \mathsf{x}_1 \exists \mathsf{x}_2 \ldots \exists \mathsf{x}_n \left(\bigwedge_{1 \leq i < j \leq n} \neg(\mathsf{x}_i = \mathsf{x}_j) \right),$$

and let

$$\Gamma' = \Gamma \cup \{\sigma_n : n \geq 2\}.$$

We claim that every finite subset of Γ' is satisfiable. Let $\Gamma'_0 \subseteq \Gamma'$ be an arbitrary finite subset of Γ. We can then fix $N \in \mathbb{N}$ such that

$$\Gamma'_0 \subseteq \Gamma \cup \{\sigma_n : 2 \leq n \leq N\}.$$

By assumption, we may fix a model $(\mathcal{M}, s)$ of Γ such that $|M| > N$. Since $|M| > N$, we have that $(\mathcal{M}, s) \vDash \sigma_n$ whenever $2 \leq n \leq N$, and hence $(\mathcal{M}, s)$ is a model of Γ'_0. Therefore, every finite subset of Γ' is satisfiable.

By the Compactness Theorem, it follows that Γ' is satisfiable. Fix a model $(\mathcal{M}', s)$ of Γ'. We then have that $(\mathcal{M}', s)$ is a model of Γ and that M' is infinite (because it is a model of σ_n for all $n \geq 2$). □

Our second proof changes the language in order to force many distinct elements.

Proof 2. Let $\mathcal{L}' = \mathcal{L} \cup \{\mathsf{c}_k : k \in \mathbb{N}\}$ where the c_k are new distinct constant symbols, and let

$$\Gamma' = \Gamma \cup \{\neg(\mathsf{c}_k = \mathsf{c}_\ell) : k, \ell \in \mathbb{N} \text{ and } k \neq \ell\}.$$

We claim that every finite subset of Γ' is satisfiable. Let $\Gamma'_0 \subseteq \Gamma'$ be an arbitrary finite subset of Γ. We can then fix $N \in \mathbb{N}$ such that

$$\Gamma'_0 \subseteq \Gamma \cup \{\neg(\mathsf{c}_k = \mathsf{c}_\ell) : k, \ell \leq N \text{ and } k \neq \ell\}.$$

By assumption, we may fix a model $(\mathcal{M}, s)$ of Γ such that $|M| > N$. Let $\mathcal{M}'$ be the $\mathcal{L}'$ structure $\mathcal{M}$ together with interpreting the constants $\mathsf{c}_0, \mathsf{c}_1, \ldots, \mathsf{c}_N$ as distinct elements of M, and interpreting each c_i for $i > N$ arbitrarily. We then have that $(\mathcal{M}', s)$ is a model of Γ'. Therefore, every finite subset of Γ' is satisfiable.

By the Compactness Theorem, it follows that Γ' is satisfiable. Fix a model $(\mathcal{M}', s)$ of Γ'. If we let $\mathcal{M}$ be the restriction of $\mathcal{M}'$ to $\mathcal{L}$, then $(\mathcal{M}, s)$ is a model of Γ which is infinite. □

As a simple application, we can show that many natural classes of finite structures are not elementary classes. Recall that if $\Sigma \subseteq Sent_{\mathcal{L}}$, then we let $Mod(\Sigma)$ be the class of all $\mathcal{L}$-structures $\mathcal{M}$ such that $\mathcal{M} \vDash \sigma$ for all $\sigma \in \Sigma$. Also recall that we say that a class $\mathcal{K}$ of $\mathcal{L}$-structures is an elementary class if there exists $\Sigma \subseteq Sent_{\mathcal{L}}$ such that $\mathcal{K} = Mod(\Sigma)$. Furthermore, $\mathcal{K}$ is a strong elementary class if we can choose a *finite* such Σ, which is equivalent to saying that we can choose Σ to consist of just one sentence (by taking the conjunction of the finitely many sentences).

Corollary 6.3.3. *The class $\mathcal{K}$ of all finite groups is not an elementary class, in either the restricted group theory language $\mathcal{L} = \{\mathsf{e}, *\}$, or the full group theory language $\mathcal{L} = \{\mathsf{e}, *, -1\}$.*

Proof. Let $\Sigma \subseteq Sent_{\mathcal{L}}$ be arbitrary such that $\mathcal{K} \subseteq Mod(\Sigma)$. We show that there is an element of $Mod(\Sigma)$ that is not in $\mathcal{K}$. Using the trivial fact that there are arbitrarily large finite groups, we know that there for every $n \in \mathbb{N}$, there exists a element of $\mathcal{K}$ with at least n elements. Therefore, for every $n \in \mathbb{N}$, there is a model of Σ with at least n elements. By Proposition 6.3.2, we conclude that there is an infinite model $\mathcal{M}$ of Σ. We then have that $\mathcal{M}$ is an element of $Mod(\Sigma)$ that is not a model of $\mathcal{K}$. □

In fact, we can greatly extend Proposition 6.3.2 to much larger structures. In this case, it is essential to follow the second proof and add lots of symbols to the language (because if $\mathcal{L}$ is countable, then every satisfiable set of formulas over $\mathcal{L}$ has a countable model by Lowenheim-Skolem).

Proposition 6.3.4. *Let $\mathcal{L}$ be a language. Suppose that $\Gamma \subseteq Form_{\mathcal{L}}$ is such that there exists a model $(\mathcal{M}, s)$ of Γ with M infinite. We then have that there exists a model $(\mathcal{M}, s)$ of Γ such that M is uncountable.*

Proof. Let $\mathcal{L}' = \mathcal{L} \cup \{\mathsf{c}_a : a \in \mathbb{R}\}$ where the c_a are new distinct constant symbols, and let

$$\Gamma' = \Gamma \cup \{\neg(\mathsf{c}_a = \mathsf{c}_b) : a, b \in \mathbb{R} \text{ and } a \neq b\}.$$

We claim that every finite subset of Γ' is satisfiable. Let $\Gamma'_0 \subseteq \Gamma'$ be an arbitrary finite subset of Γ'. We can then fix a finite $Z \subseteq \mathbb{R}$ such that

$$\Gamma'_0 \subseteq \Gamma \cup \{\neg(\mathsf{c}_a = \mathsf{c}_b) : a, b \in Z\}.$$

By assumption, we may fix a model $(\mathcal{M}, s)$ of Γ such that M is infinite. Let $\mathcal{M}'$ be the $\mathcal{L}'$ structure $\mathcal{M}$ together with interpreting the constants c_a for $a \in Z$ as distinct elements of M, and interpreting each c_b for $b \notin Z$ arbitrarily. We then have that $(\mathcal{M}', s)$ is a model of Γ'. Hence, every finite subset of Γ' is satisfiable.

By the Compactness Theorem, it follows that Γ' is satisfiable. Fix a model $(\mathcal{M}', s)$ of Γ'. If we let $\mathcal{M}$ be the restriction of $\mathcal{M}'$ to $\mathcal{L}$, then $(\mathcal{M}, s)$ is a model of Γ which is uncountable. □

By using the idea of adding a special new constant symbol to our language, we can show that other natural classes are not elementary classes. As an example, consider the class of all *torsion* groups, i.e. the class of groups in which every element has finite order.

Proposition 6.3.5. *The class $\mathcal{K}$ of all torsion groups is not an elementary class, in either the restricted group theory language $\mathcal{L} = \{\mathsf{e}, *\}$, or the full group theory language $\mathcal{L} = \{\mathsf{e}, *, -1\}$.*

Proof. Let $\Sigma \subseteq Sent_{\mathcal{L}}$ be arbitrary such that $\mathcal{K} \subseteq Mod(\Sigma)$. Let $\mathcal{L}' = \mathcal{L} \cup \{\mathsf{c}\}$ where c is a new constant symbol. For each $n \in \mathbb{N}^+$, let $\tau_n \in Sent_{\mathcal{L}'}$ be $\neg(\mathsf{c}^n = \mathsf{e})$. More formally, for each $n \in \mathbb{N}^+$, we let τ_n be the sentence $\neg(\mathsf{c} * \mathsf{c} * \cdots * \mathsf{c} = \mathsf{e})$, where there are n many c's. Now let

$$\Sigma' = \Sigma \cup \{\tau_n : n \in \mathbb{N}^+\}.$$

We claim that every finite subset of Σ' has a model. Let $\Sigma'_0 \subseteq \Sigma'$ be an arbitrary finite subset of Σ'. Fix $N \in \mathbb{N}$ such that

$$\Sigma'_0 \subseteq \Sigma \cup \{\tau_n : n < N\}.$$

Notice that if we let $\mathcal{M}'$ be the group $\mathbb{Z}/N\mathbb{Z}$ and let $\mathsf{c}^{\mathcal{M}'} = \overline{1}$, then $\mathcal{M}'$ is a model of Σ_0'. Thus, every finite subset of Σ' has a model, so Σ' has a model by Compactness. If we restrict this model to $\mathcal{L}$, we obtain a structure $\mathcal{M}$ in $Mod(\Sigma)$ which is not in $\mathcal{K}$ because it has an element (namely $\mathsf{c}^{\mathcal{M}'}$) of infinite order. Therefore, $\mathcal{K} \neq Mod(\Sigma)$. □

Proposition 6.3.6. *The class $\mathcal{K}$ of all equivalence relations in which all equivalence classes are finite is not an elementary class in the language $\mathcal{L} = \{\mathsf{R}\}$.*

Proof. Suppose that $\Sigma \subseteq Sent_{\mathcal{L}}$ is such that $\mathcal{K} \subseteq Mod(\Sigma)$. Let $\mathcal{L}' = \mathcal{L} \cup \{\mathsf{c}\}$ where c is new constant symbol. For each $n \in \mathbb{N}^+$, let $\tau_n \in Sent_{\mathcal{L}'}$ be

$$\exists \mathsf{x}_1 \exists \mathsf{x}_2 \cdots \exists \mathsf{x}_n \left(\bigwedge_{1 \leq i < j \leq n} (\mathsf{x}_i \neq \mathsf{x}_j) \wedge \bigwedge_{i=1}^{n} \mathsf{R}\mathsf{c}\mathsf{x}_i \right)$$

and let

$$\Sigma' = \Sigma \cup \{\tau_n : n \in \mathbb{N}\}.$$

We claim that every finite subset of Σ' has a model. Let $\Sigma_0' \subseteq \Sigma'$ be an arbitrary finite subset of Σ'. Fix $N \in \mathbb{N}$ such that

$$\Sigma_0 \subseteq \Sigma \cup \{\tau_n : n \leq N\}.$$

Notice that if we let $M' = \{0, 1, 2, \ldots, N\}$, $\mathsf{R}^{\mathcal{M}'} = (M')^2$, and $\mathsf{c}^{\mathcal{M}'} = 0$, then $\mathcal{M}'$ is a model of Σ_0'. Thus, every finite subset of Σ' has a model, so Σ' has a model by Compactness. If we restrict this model to $\mathcal{L}$, we get an element of $Mod(\Sigma)$ which is not in $\mathcal{K}$ because it has an infinite equivalence class. □

We can also use the Compactness Theorem to show that certain elementary classes are not strong elementary classes. The key result behind such arguments is the following proposition, which says that if we have a strong elementary class $\mathcal{K}$ that we have already know is equal to $Mod(\Sigma)$ for an infinite set Σ, then we can find a finite subset of Σ itself witnessing the fact that that $\mathcal{K}$ is a strong elementary class.

Proposition 6.3.7. *Suppose that $\mathcal{K}$ is a strong elementary class, that $\Sigma \subseteq Sent_{\mathcal{L}}$, and that $\mathcal{K} = Mod(\Sigma)$. There exists a finite $\Sigma_0 \subseteq \Sigma$ such that $\mathcal{K} = Mod(\Sigma_0)$.*

Proof. Since $\mathcal{K}$ is a strong elementary class, we may fix $\tau \in Sent_{\mathcal{L}}$ with $\mathcal{K} = Mod(\tau)$. We then have $\Sigma \vDash \tau$ because any model of Σ is an element of $Mod(\Sigma) = \mathcal{K} = Mod(\tau)$, and hence is a model of τ. Therefore, by Compactness we may fix a finite $\Sigma_0 \subseteq \Sigma$ such that $\Sigma_0 \vDash \tau$. Now notice that $\mathcal{K} = Mod(\Sigma) \subseteq Mod(\Sigma_0)$ and $Mod(\Sigma_0) \subseteq Mod(\tau) = \mathcal{K}$, so $\mathcal{K} = Mod(\Sigma_0)$. □

Corollary 6.3.8. *The class $\mathcal{K}$ of all fields of characteristic 0 is an elementary class, but not a strong elementary class, in the ring theory language $\mathcal{L} = \{\mathsf{0}, \mathsf{1}, +, -, \cdot\}$.*

Proof. We already know that $\mathcal{K}$ is an elementary class because if we let σ be the conjunction of the fields axioms and let τ_n be $\mathsf{1} + \mathsf{1} + \cdots + \mathsf{1} \neq \mathsf{0}$ (where there are n many $\mathsf{1}$'s) for each $n \in \mathbb{N}^+$, then $\mathcal{K} = Mod(\Sigma)$ where

$$\Sigma = \{\sigma\} \cup \{\tau_n : n \in \mathbb{N}^+\}.$$

Assume then that $\mathcal{K}$ is a strong elementary class. By Proposition 6.3.7, we may fix a finite $\Sigma_0 \subseteq \Sigma$ such that $\mathcal{K} = Mod(\Sigma_0)$. Fix $N \in \mathbb{N}$ such that

$$\Sigma_0 \subseteq \{\sigma\} \cup \{\tau_n : n \leq N\}.$$

Now if fix a prime $p > N$ (which is possible because there are infinitely many primes) we see that $(\mathbb{Z}/p\mathbb{Z}, 0, 1, +, \cdot)$ is a model of Σ_0 which is not an element of $\mathcal{K}$. This is a contradiction, so $\mathcal{K}$ is not a strong elementary class. □

For each prime p, there is a field with p elements, namely the ring $\mathbb{Z}/p\mathbb{Z}$. For the remainder of this section, we use the notation $\mathbb{F}_p$ to denote this field. Recall that every field F can be embedded in an algebraically closed field, and in fact, there is a unique (up to isomorphism) algebraically closed field K extending F that is an algebraic extension of F. Such a field K is called an algebraic closure of F. Since this field is unique up isomorphism, we denote any particular algebraic closure of F by the notation $\overline{F}$.

Theorem 6.3.9. *Let $\mathcal{L} = \{0, 1, +, \cdot, -\}$ be the language of rings, and let $\sigma \in Sent_{\mathcal{L}}$. The following are equivalent:*

1. *$ACF_0 \vDash \sigma$ (which is equivalent to $\sigma \in ACF_0$ because ACF_0 is a theory).*
2. *$\mathbb{C} \vDash \sigma$.*
3. *There exists $m \in \mathbb{N}$ such that $ACF_p \vDash \sigma$ for all primes $p > m$.*
4. *There exists $m \in \mathbb{N}$ such that $\overline{\mathbb{F}}_p \vDash \sigma$ for all primes $p > m$.*
5. *$ACF_p \vDash \sigma$ for infinitely many primes p.*
6. *$\overline{\mathbb{F}}_p \vDash \sigma$ for infinitely many primes p.*

Proof. Recall from Corollary 5.6.6 that ACF_0 is complete. We claim that this implies that (1) and (2) are equivalent. To see this, notice first that if $ACF_0 \vDash \sigma$, then $\mathbb{C} \vDash \sigma$ because $\mathbb{C}$ is a model of ACF_0. For the converse, notice that if $ACF_0 \nvDash \sigma$, then $ACF_0 \vDash \neg\sigma$ because ACF_0 is complete, so $\mathbb{C} \vDash \neg\sigma$ and hence $\mathbb{C} \nvDash \sigma$. Similarly, since each ACF_p is complete by Corollary 5.6.6, and since each $\overline{\mathbb{F}}_p$ is a model of ACF_p, we obtain the equivalence of (3) and (4), along with the equivalence of (5) and (6). Next notice that (3) implies (5) trivially. To complete the proof, we show that (1) implies (3) and that (5) implies (1).

First, we show that (1) implies (3). Suppose then that $ACF_0 \vDash \sigma$. For each $n \in \mathbb{N}^+$, let τ_n be the sentence saying that $\mathsf{1}$ added to itself n times does not equal $\mathsf{0}$, and let ρ_n be the sentence saying that every polynomial of degree n with nonzero leading coefficient has a root (see the beginning of Section 5.6 for the formal sentences). Finally, let π be the conjunction of the field axioms, and let

$$\Sigma = \{\pi\} \cup \{\rho_n : n \in \mathbb{N}^+\} \cup \{\tau_n : n \in \mathbb{N}^+\}.$$

We then have that $ACF_0 = Cn(\Sigma)$, so since $ACF_0 \vDash \sigma$, it follows that $\Sigma \vDash \sigma$. By Compactness, we can fix a finite $\Sigma_0 \subseteq \Sigma$ such that $\Sigma_0 \vDash \tau$. Fix an $N \in \mathbb{N}$ such that

$$\Sigma_0 \subseteq \{\pi\} \cup \{\rho_n : n \in \mathbb{N}^+\} \cup \{\tau_n : n \leq N\}.$$

Now if p is any prime greater than N, then any algebraically closed field of characteristic p is a model of Σ_0, and hence is a model of τ. Therefore, $ACF_p \vDash \sigma$ for all $p > N$.

We finally show that (5) implies (1) by proving the contrapositive. Suppose then that $ACF_0 \nvDash \sigma$. Since ACF_0 is complete, we have that $ACF_0 \vDash \neg\sigma$. Since we have already established that (1) implies (3), we can fix $m \in \mathbb{N}$ such that $ACF_p \vDash \neg\sigma$ for all primes $p > m$. Since each ACF_p is satisfiable, it follows that $ACF_p \nvDash \sigma$ for all primes $p > m$. Therefore, the set of primes p such that $ACF_p \vDash \sigma$ is finite. □

Before using this theorem to prove an amazing result, we first summarize several important facts about finite fields. Recall that every finite field has characteristic equal to some prime p. Furthermore, if F has characteristic p, then F contains a copy of $\mathbb{Z}/p\mathbb{Z}$. From here, it follows that a finite field of characteristic p can be viewed as a vector space of $\mathbb{Z}/p\mathbb{Z}$, and hence has p^n many elements for some n (because if we fix a basis of F over $\mathbb{Z}/p\mathbb{Z}$, then there is a finite number n of elements of the basis, from which we can describe elements of F uniquely by picking n coefficients from $\mathbb{Z}/p\mathbb{Z}$). Next, if F is a field of characteristic p, and $a, b \in F$, then $(a + b)^p = a^p + b^p$ for all $a, b \in F$ (essentially this comes from the fact that all of the other

coefficients of binomial expansion is divisible by p, and hence equal 0 in F). By induction, it follows that for all $a, b \in F$ and $n \in \mathbb{N}^+$, we have

$$(a+b)^{p^n} = a^{p^n} + b^{p^n}.$$

Another important fact is that if F is a finite field with $|F| = p^n$ (and hence of characteristic p), then we have

$$a^{p^n - 1} = 1$$

for all nonzero $a \in F$ by Lagrange's Theorem (because $F \backslash \{0\}$ is a finite group of order $p^n - 1$ under multiplication). Therefore, we have

$$a^{p^n} = a$$

for all $a \in F$, including 0. With all of this in mind, we have the following fact.

Proposition 6.3.10. *Let p be prime. Every finitely generated subfield of $\overline{\mathbb{F}}_p$ is finite.*

Proof. Let p be an arbitrary prime. For each $n \in \mathbb{N}^+$, let $K_n = \{a \in \overline{\mathbb{F}}_p : a^{p^n} = a\}$, and notice that K_n is the set of roots of $x^{p^n} - x$ in $\overline{\mathbb{F}}_p$. Since $\overline{\mathbb{F}}_p$ is algebraically closed, we know that $x^{p^n} - x$ splits in $\overline{\mathbb{F}}_p$. Notice that $x^{p^n} - x$ is a separable polynomial because it has formal derivative equal to -1, which is trivially relatively prime to $x^{p^n} - x$. Therefore, the roots of $x^{p^n} - x$ in $\overline{\mathbb{F}}_p$ are distinct, and hence $|K_n| = p^n$. Furthermore, we have that each K_n is closed under addition and multiplication because if $a, b \in K_n$, then

$$(a+b)^{p^n} = a^{p^n} + b^{p^n} = a + b$$

from above, and

$$(ab)^{p^n} = a^{p^n} b^{p^n} = ab.$$

Notice that $-1 \in K_n$ because we have $(-1)^{p^n} = -1$ whenever p is odd, and $(-1)^{p^n} = 1 = -1$ when $p = 2$. Since K_n is closed under multiplication, it follows that if $a \in K_n$ is arbitrary, then $-a = (-1) \cdot a \in K_n$. Finally, if if $a \in K$ is nonzero, then

$$(a^{-1})^{p^n} = (a^{p^n})^{-1} = a^{-1},$$

so K_n is closed under multiplicative inverses (of nonzero elements). Therefore, K_n is a subfield of $\overline{\mathbb{F}}_p$ with p^n elements. Moreover, if L is an arbitrary subfield of $\overline{\mathbb{F}}_p$ with p^n many elements, then $a^{p^n} = a$ for all $a \in L$ from above, so $L \subseteq K$, and hence $L = K$ because L and K are finite sets of the same cardinality. In other words, K_n is the unique subfield of $\overline{\mathbb{F}}_p$ with p^n elements.

Next, we claim that if $d \mid n$, then $K_d \subseteq K_n$. To see this, let $a \in K_d$ be arbitrary. We then have $a^{p^d} = a$, so

$$a^{p^{2 \cdot d}} = (a^{p^d})^{p^d} = a^{p^d} = a$$

hence

$$a^{p^{3 \cdot d}} = (a^{p^{2 \cdot d}})^{p^d} = a^{p^d} = a.$$

By a simple induction, it follows that $a^{p^{m \cdot d}} = a$ for all $m \in \mathbb{N}^+$. Thus, if $d \mid n$, then $K_d \subseteq K_n$.

Let $K = \bigcup_{n \in \mathbb{N}} K_n$. We claim that $K = \overline{\mathbb{F}}_p$. We clearly have that $K \subseteq \overline{\mathbb{F}}_p$. To show equality, it suffices to show that K itself is algebraically closed. Let $f(x) \in K[x]$ be an arbitrary nonconstant polynomial, and write $f(x) = a_m x^m + a_{m-1} x^{m-1} + \cdots + a_1 x + a_0$ where each $a_i \in K$. Using the fact that $K = \bigcup_{n \in \mathbb{N}} K_n$ and that $K_d \subseteq K_n$ whenever $d \mid n$, there exists $N \in \mathbb{N}^+$ such that $a_i \in K_N$ for all i (by taking a least common multiple). Now $f(x) \in K_N[x] \subseteq \overline{\mathbb{F}}_p[x]$, so since $\overline{\mathbb{F}}_p$ is algebraically closed, we know that it has some root α of $f(x)$. We now have $K_N \subseteq K_N(\alpha) \subseteq \overline{\mathbb{F}}_p$, and that $K_N(\alpha)$ is a finite extension of K_N. Since K_N is a finite field, and $K_N(\alpha)$ is a finite extension of K_N, the field $K_N(\alpha)$ is a finite subfield of $\overline{\mathbb{F}}_p$. Since every finite subfield of $\overline{\mathbb{F}}_p$ equals some K_n, we conclude that $K_N(\alpha) \subseteq K$, and hence $\alpha \in K$. Therefore, every polynomial over K has a root in K, and hence K is algebraically closed. It follows that $K = \overline{\mathbb{F}}_p$.

We now prove the result. Let $a_1, a_2, \ldots, a_m \in \overline{\mathbb{F}}_p$ be arbitrary. Since $\overline{\mathbb{F}}_p = K$, we have $a_1, a_2, \ldots, a_m \in K$. By taking a least common multiple, we can fix $N \in \mathbb{N}^+$ such that $a_1, a_2, \ldots, a_m \in K_N$. We then have that the subfield of $\overline{\mathbb{F}}_p$ generated by $a_1, a_2, \ldots, a_m$ is a subfield of K_N, and hence is finite. $\square$

Given a field F, each polynomial in $F[x]$ determines a function from F to F via evaluation. Similarly, an element of $F[x_1, x_2, \ldots, x_n]$, i.e. a polynomial of several variables, determines a function from F^n to F via evaluation. By putting several such polynomial functions (in the same number of variables) together, we can define polynomial functions from F^n to F^m. For example, if $f_1(x, y) = x^2y + 5x$ and $f_2(x, y) = x^5y - 3xy^2 + 7y^3$, then $f(x, y) = (f_1(x, y), f_2(x, y))$ can be viewed as a polynomial function from $\mathbb{Q}^2$ to $\mathbb{Q}^2$ (or really from F^2 to F^2 for any field F).

Theorem 6.3.11 (Ax-Grothendieck)**.** *Every injective polynomial map from $\mathbb{C}^n$ to $\mathbb{C}^n$ is surjective.*

Proof. Let $\mathcal{L} = \{0, 1, +, -, \cdot\}$ be the language of rings. Notice that given any $n, d \in \mathbb{N}^+$, we can write a sentence $\sigma_{n,d} \in Sent_{\mathcal{L}}$ expressing that every injective polynomial map from F^n to F^n, where each polynomial has degree at most d, is surjective. We want to show that $\mathbb{C} \vDash \sigma_{n,d}$ for all $n, d \in \mathbb{N}^+$. By Theorem 6.3.9, it suffices to show that $\overline{\mathbb{F}}_p \vDash \sigma_{n,d}$ for all primes p and all $n, d \in \mathbb{N}^+$. Thus, it suffices to show that for all primes p and all $n \in \mathbb{N}^+$, every injective polynomial map from $\overline{\mathbb{F}}_p^n$ to $\overline{\mathbb{F}}_p^n$ is surjective.

Let $p, n \in \mathbb{N}^+$ be arbitrary with p prime. Let $f\colon \overline{\mathbb{F}}_p^n \to \overline{\mathbb{F}}_p^n$ be an arbitrary injective polynomial map, and let $(b_1, b_2, \ldots, b_n) \in \overline{\mathbb{F}}_p^n$ be arbitrary. We need to show that there exists $(a_1, a_2, \ldots, a_n) \in \overline{\mathbb{F}}_p^n$ with $f(a_1, a_2, \ldots, a_n) = (b_1, b_2, \ldots, b_n)$. Let $f_1, f_2, \ldots, f_n \in \overline{\mathbb{F}}_p[x_1, x_2, \ldots, x_n]$ be such that $f = (f_1, f_2, \ldots, f_n)$, and let C be the finite set of coefficients appearing in $f_1, f_2, \ldots, f_n$. Let K be the subfield of $\overline{\mathbb{F}}_p$ generated by $C \cup \{b_1, b_2, \ldots, b_n\}$ and notice that K is a finite field by Proposition 6.3.10. Now $f \restriction K^n$ maps K^n into K^n and is injective, so must be surjective because K^n is finite. Thus, there exists $(a_1, a_2, \ldots, a_n) \in K^n \subseteq \overline{\mathbb{F}}_p^n$ such that $f(a_1, a_2, \ldots, a_n) = (b_1, b_2, \ldots, b_n)$. □

6.4 Random Graphs

Throughout this section, we work in the language $\mathcal{L} = \{\mathsf{R}\}$ where R is binary relation symbol. We consider loopless undirected graphs, which we view as $\mathcal{L}$-structures that are models of $\{\forall \mathsf{x} \neg \mathsf{Rxx}, \forall \mathsf{x} \forall \mathsf{y} (\mathsf{Rxy} \to \mathsf{Ryx})\}$.

Definition 6.4.1. *For each $n \in \mathbb{N}^+$, let $\mathcal{G}_n$ be the set of of all models of $\{\forall \mathsf{x} \neg \mathsf{Rxx}, \forall \mathsf{x} \forall \mathsf{y} (\mathsf{Rxy} \to \mathsf{Ryx})\}$ with universe $[n]$.*

Definition 6.4.2. *For each $\mathcal{A} \subseteq \mathcal{G}_n$, we let*

$$Pr_n(\mathcal{A}) = \frac{|\mathcal{A}|}{|\mathcal{G}_n|}.$$

For each $\sigma \in Sent_{\mathcal{L}}$, we let

$$Pr_n(\sigma) = \frac{|\{\mathcal{M} \in \mathcal{G}_n : \mathcal{M} \vDash \sigma\}|}{|\mathcal{G}_n|}.$$

We use the suggestive Pr because we think of constructing a graph randomly by flipping a fair coin for each 2-element subset $\{i, j\}$ of $[n]$ to determine whether or not there is an edge linking them. In this context, $Pr_n(\mathcal{A})$ is the probability the graph so constructed with vertex set $[n]$ is an element of $\mathcal{A}$. Notice that if $\mathcal{A}$ and $\mathcal{B}$ are both subsets of $\mathcal{G}_n$, then we trivially have $|\mathcal{A} \cup \mathcal{B}| \leq |\mathcal{A}| + |\mathcal{B}|$, and hence

$$\begin{aligned} Pr_n(\mathcal{A} \cup \mathcal{B}) &= \frac{|\mathcal{A} \cup \mathcal{B}|}{|\mathcal{G}_n|} \\ &\leq \frac{|\mathcal{A}| + |\mathcal{B}|}{|\mathcal{G}_n|} \\ &\leq \frac{|\mathcal{A}|}{|\mathcal{G}_n|} + \frac{|\mathcal{B}|}{|\mathcal{G}_n|} \\ &= Pr_n(\mathcal{A}) + Pr_n(\mathcal{B}). \end{aligned}$$

More generally if $\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_k$ are all subsets of $\mathcal{G}_n$, then

$$Pr_n(\mathcal{A}_1 \cup \mathcal{A}_2 \cup \cdots \cup \mathcal{A}_k) \leq Pr_n(\mathcal{A}_1) + Pr_n(\mathcal{A}_2) + \cdots + Pr_n(\mathcal{A}_k).$$

Notice that if $\mathcal{A} \subseteq \mathcal{B} \subseteq \mathcal{G}_n$, then

$$\begin{aligned} Pr_n(\mathcal{A}) &= \frac{|\mathcal{A}|}{|\mathcal{G}_n|} \\ &\leq \frac{|\mathcal{B}|}{|\mathcal{G}_n|} \\ &= Pr_n(\mathcal{B}) \end{aligned}$$

and for any $\mathcal{A} \subseteq \mathcal{G}_n$, we have

$$Pr_n(\mathcal{G}_n \backslash \mathcal{A}) = 1 - Pr_n(\mathcal{A}).$$

Now given given two distinct 2-element subsets $\{i, j\}$ and $\{i', j'\}$ of $\{1, 2, \ldots, n\}$, the question of whether there is an edge linking i and j and the question of whether there is an edge linking i' and j' is independent. More formally, if

$$\mathcal{A}_1 = \{\mathcal{M} \in \mathcal{G}_n : (i, j) \in \mathsf{R}^{\mathcal{M}}\}$$

and

$$\mathcal{A}_2 = \{\mathcal{M} \in \mathcal{G}_n : (i', j') \in \mathsf{R}^{\mathcal{M}}\},$$

then we have

$$Pr_n(\mathcal{A}_1 \cap \mathcal{A}_2) = \frac{1}{4} = \frac{1}{2} \cdot \frac{1}{2} = Pr_n(\mathcal{A}_1) \cdot Pr_n(\mathcal{A}_2).$$

More generally, suppose that we have sets $E_1, E_2, \ldots, E_k$, where each E_i is a set of 2-element subsets of $[n]$. Suppose that we fix choices for whether each pair in E_i should be in the graph or not, and let $\mathcal{A}_i$ be the subset of $\mathcal{G}_n$ consisting of those edges that meet the requirements for the edges in E_i. If the sets $E_1, E_2, \ldots, E_k$ are pairwise disjoint, then

$$Pr_n(\mathcal{A}_1 \cap \mathcal{A}_2 \cap \cdots \cap \mathcal{A}_k) = Pr_n(\mathcal{A}_1) \cdot Pr_n(\mathcal{A}_2) \cdots Pr_n(\mathcal{A}_k).$$

In other words, the events saying that the edges in each of E_i behave according to a specified pattern are mutually independent when the E_i are pairwise disjoint.

For example, suppose that σ is the sentence

$$\exists \mathsf{x} \exists \mathsf{y} \exists \mathsf{z} (\mathsf{Rxy} \wedge \mathsf{Ryz} \wedge \mathsf{Rzx})$$

saying that the graph has a triangle. We want to understand $Pr_n(\sigma)$. Determining the exact values for various n is difficult, but we can classify the long term behavior as n gets large. To see this, let $n \in \mathbb{N}^+$ be arbitrary. Consider partitioning the n vertices of $[n]$ into $\lfloor \frac{n}{3} \rfloor$ many set of size 3 (with perhaps 1 or 2 vertices left over) by considering the sets $\{1, 2, 3\}$, then $\{4, 5, 6\}$, etc. Now for any one of these sets, the probability that the vertices in the set forms a triangle is $(\frac{1}{2})^3 = \frac{1}{8}$, so the probability that the vertices in a set do not form a triangle is $1 - \frac{1}{8} = \frac{7}{8}$. Since these events are mutually independent (as the corresponding potential edge sets are disjoint), the probability that none of these families forms a triangle equals

$$\left(\frac{7}{8}\right)^{\lfloor \frac{n}{3} \rfloor}.$$

Therefore, the probability that there is *no* triangle, which is $Pr_n(\neg\sigma)$, satisfies the following inequality:

$$Pr_n(\neg\sigma) \leq \left(\frac{7}{8}\right)^{\lfloor \frac{n}{3} \rfloor} \leq \left(\frac{7}{8}\right)^{\frac{n}{3}}.$$

It follows that

$$\begin{aligned} Pr_n(\sigma) &= 1 - Pr_n(\neg\sigma) \\ &\geq 1 - \left(\frac{7}{8}\right)^{\frac{n}{3}}. \end{aligned}$$

Since

$$\lim_{n\to\infty} \left(1 - \left(\frac{7}{8}\right)^{\frac{n}{3}}\right) = 1 - 0 = 1,$$

and we trivially have $Pr_n(\sigma) \leq 1$ for all $n \in \mathbb{N}^+$, we conclude that $\lim\limits_{n\to\infty} Pr_n(\sigma) = 1$. In other words, when n is large, a randomly constructed graph on $[n]$ will almost surely have a triangle.

For another example, consider the sentence

$$\forall \mathsf{x}_1 \forall \mathsf{x}_2 (\neg(\mathsf{x}_1 = \mathsf{x}_2) \to \exists \mathsf{z}(\neg(\mathsf{z} = \mathsf{x}_1) \wedge \neg(\mathsf{z} = \mathsf{x}_2) \wedge \mathsf{R}\mathsf{x}_1\mathsf{z} \wedge \mathsf{R}\mathsf{x}_2\mathsf{z}))$$

saying that whenever we have two distinct vertices, we can find a common neighbor. Let $n \in \mathbb{N}$ with $n \geq 3$ be arbitrary. Consider arbitrary distinct $a_1, a_2 \in [n]$. For each $c \in [n]$ distinct from the a_1 and a_2, let

$$\mathcal{A}_c = \{\mathcal{M} \in \mathcal{G}_n : c \text{ is adjacent to both } a_1 \text{ and } a_2\},$$

so that

$$\mathcal{G}_n \backslash \mathcal{A}_c = \{\mathcal{M} \in \mathcal{G}_n : c \text{ is not adjacent to at least one of } a_1 \text{ or } a_2\}.$$

For each such c, we have $Pr_n(\mathcal{A}_c) = (\frac{1}{2})^2 = \frac{1}{4}$, so $Pr_n(\mathcal{G}_n \backslash \mathcal{A}_c) = 1 - \frac{1}{4} = \frac{3}{4}$. As we vary c through the $n-2$ other vertices, the corresponding events $\mathcal{G}_n \backslash \mathcal{A}_c$ are mutually independent, so the probability that no c is a common neighbor for this particular pair $\{a_1, a_2\}$ equals

$$Pr_n\left(\bigcap_c (\mathcal{G}_n \backslash \mathcal{A}_c)\right) = \prod_c Pr_n(\mathcal{G}_n \backslash \mathcal{A}_c) = \left(\frac{3}{4}\right)^{n-2}.$$

Now there are $\binom{n}{2}$ possible pairs of distinct vertices a_1 and a_2, so the probability that there exists such a pair with no common neighbor is

$$\begin{aligned} Pr_n(\neg\sigma) &\leq \binom{n}{2} \cdot \left(\frac{3}{4}\right)^{n-2} \\ &= \frac{n(n-1)}{2} \cdot \left(\frac{3}{4}\right)^{n-2}. \end{aligned}$$

Since

$$\lim_{n\to\infty} \frac{n(n-1)}{2} \cdot \left(\frac{3}{4}\right)^{n-2} = 0$$

(see the proof of Proposition 6.4.5), it follows that $\lim\limits_{n\to\infty} Pr(\neg\sigma) = 0$. Using the fact that $Pr_n(\sigma) = 1 - Pr_n(\neg\sigma)$, we conclude that $\lim\limits_{n\to\infty} Pr_n(\sigma) = 1$.

We aim to prove the following result, originally proven by Glebskii, Kogan, Liagonkii, and Talanov, but also independently by Fagin.

Theorem 6.4.3. *For all $\sigma \in Sent_{\mathcal{L}}$, either $\lim\limits_{n\to\infty} Pr_n(\sigma) = 1$ or $\lim\limits_{n\to\infty} Pr_n(\sigma) = 0$.*

The key step in proving this result is generalizing the last example.

Definition 6.4.4. *For each $r, s \in \mathbb{N}$ with $\max\{r, s\} > 0$, let $\sigma_{r,s}$ be the sentence*

$$\forall x_1 \forall x_2 \cdots \forall x_r \forall y_1 \forall y_2 \cdots \forall y_s \Big(\bigwedge_{1 \le i < j \le r} (x_i \neq x_j) \wedge \bigwedge_{1 \le i < j \le s} (y_i \neq y_j) \wedge \bigwedge_{i=1}^{r} \bigwedge_{j=1}^{s} (x_i \neq y_j) \Big)$$
$$\rightarrow \exists z \Big(\bigwedge_{i=1}^{r} (z \neq x_i) \wedge \bigwedge_{j=1}^{s} (z \neq y_j) \wedge \bigwedge_{i=1}^{r} Rx_i z \wedge \bigwedge_{j=1}^{s} \neg Ry_j z \Big) \Big)$$

If $s = 0$, we simply omit all quantifiers and conjuncts mentioning a y_j. Similarly, if $r = 0$, we simply omit all quantifiers and conjuncts mentioning a x_i.

Intuitively, a graph is model of $\sigma_{r,s}$ if it has the property that whenever A and B are disjoint sets of vertices with $|A| = r$ and $|B| = s$, we can always find a vertex $u \notin A \cup B$ that is adjacent to every element of A, but not adjacent to any element of B. Peter Winkler has called the statements $\sigma_{r,s}$ the *Alice's Restaurant* axioms, references Arlo Guthrie's story/song containing the line "You can get anything you want at Alice's Restaurant".

Proposition 6.4.5. *For all $r, s \in \mathbb{N}$ with $\max\{r, s\} > 0$, we have $\lim_{n \to \infty} Pr_n(\sigma_{r,s}) = 1$.*

Proof. Let $r, s \in \mathbb{N}$ be arbitrary with $\max\{r, s\} > 0$. Let $n \in \mathbb{N}$ be arbitrary with $n > r + s$. Consider arbitrary disjoint subsets U and W of $[n]$ with $|U| = r$ and $|W| = s$. For each $c \in [n] \backslash (U \cup W)$, let

$$\mathcal{A}_c = \{\mathcal{M} \in \mathcal{G}_n : c \text{ is adjacent to each element of } U \text{ and to no element of } W\}$$

For each such c, we have $Pr_n(\mathcal{A}_c) = \frac{1}{2^{r+s}}$, so $Pr_n(\mathcal{G}_n \backslash \mathcal{A}_c) = 1 - \frac{1}{2^{r+s}}$. As we vary c through the $n - r - s$ vertices in $[n] \backslash (U \cup W)$, the corresponding events $\mathcal{G}_n \backslash \mathcal{A}_c$ are mutually independent, so the probability that no c works for this choice of U and W equals

$$\left(1 - \frac{1}{2^{r+s}}\right)^{n-r-s}.$$

Now there are $\binom{n}{r} \cdot \binom{n-r}{s}$ possible choices for the sets U and W, so

$$\begin{aligned} Pr_n(\neg\sigma_{r,s}) &\le \binom{n}{r}\binom{n-r}{s}\left(1 - \frac{1}{2^{r+s}}\right)^{n-r-s} \\ &\le n^r \cdot n^s \cdot \left(1 - \frac{1}{2^{r+s}}\right)^{n-r-s} \\ &= \left(1 - \frac{1}{2^{r+s}}\right)^{-r-s} \cdot n^{r+s} \cdot \left(1 - \frac{1}{2^{r+s}}\right)^{n} \\ &= \left(1 - \frac{1}{2^{r+s}}\right)^{-r-s} \cdot n^{r+s} \cdot \left(\frac{2^{r+s}-1}{2^{r+s}}\right)^{n} \\ &= \left(1 - \frac{1}{2^{r+s}}\right)^{-r-s} \cdot \frac{n^{r+s}}{\left(\frac{2^{r+s}}{2^{r+s}-1}\right)^{n}} \end{aligned}$$

Now we use the fact that if $k \in \mathbb{N}^+$ and $r \in \mathbb{R}$ with $r > 1$, then

$$\lim_{n \to \infty} \frac{n^k}{r^n} = 0$$

(i.e. that any exponential eventually dominates any polynomial) to conclude that $\lim_{n \to \infty} Pr_n(\neg\sigma_{r,s}) = 0$. Therefore $\lim_{n \to \infty} Pr_n(\sigma_{r,s}) = 1$. □

Definition 6.4.6. *Let* $\Sigma = \{\forall \mathsf{x} \neg \mathsf{Rxx}, \forall \mathsf{x} \forall \mathsf{y}(\mathsf{Rxy} \to \mathsf{Ryx})\} \cup \{\sigma_{r,s} : r, s \in \mathbb{N}^+ \text{ and } \max\{r, s\} > 0\}$ *and let* $RG = Cn(\Sigma)$.

Proposition 6.4.7. *RG is satisfiable.*

Proof. We build a countable model $\mathcal{M}$ of RG with $M = \mathbb{N}$. Notice first that since $\mathcal{P}_{fin}(\mathbb{N})$ (the set of all finite subsets of $\mathbb{N}$) is countable, so is the set $\mathcal{P}_{fin}(\mathbb{N})^2$, Hence the set

$$\{(A, B) \in \mathcal{P}_{fin}(\mathbb{N})^2 : A \cap B = \emptyset \text{ and } A \cup B \neq \emptyset\}$$

is countable. Therefore, we may list it as

$$(A_1, B_1), (A_2, B_2), (A_3, B_3), \ldots$$

and furthermore we may assume that $\max(A_n \cup B_n) < n$ for all $n \in \mathbb{N}$. Let $\mathcal{M}$ be the $\mathcal{L}$-structure where $M = \mathbb{N}$ and $\mathsf{R}^{\mathcal{M}} = \{(k, n) : k \in A_n\} \cup \{(n, k) : k \in A_n\}$. Suppose now that $A, B \subseteq \mathbb{N}$ are finite with $A \cap B = \emptyset$ and $A \cup B \neq \emptyset$. Fix $n \in \mathbb{N}$ with $A = A_n$ and $B = B_n$. We then have that $(k, n) \in \mathsf{R}^{\mathcal{M}}$ for all $k \in A$ (because $k \in A_n$) and $(\ell, n) \notin \mathsf{R}^{\mathcal{M}}$ for all $\ell \in B$ (because $\ell \notin A_n$ and $n \notin A_\ell$ since $\ell < n$). Therefore, $\mathcal{M} \vDash \sigma_{r,s}$ for all $r, s \in \mathbb{N}$ with $\max r, s > 0$. Thus, $\mathcal{M}$ is a model of RG. □

Theorem 6.4.8. *All models of RG are infinite, and any two countable models of RG are isomorphic.*

Proof. Let $\mathcal{M}$ be an arbitrary model of RG. Suppose that $\mathcal{M}$ is finite, and let $n = |M|$. Since $\mathcal{M} \vDash \sigma_{n,0}$, there exists $b \in M$ such that $(b, a) \in \mathsf{R}^{\mathcal{M}}$ for all $a \in M$. However, this is a contradiction because $(a, a) \notin \mathsf{R}^{\mathcal{M}}$ for all $a \in M$. It follows that all models of RG are infinite.

Suppose now that $\mathcal{M}$ and $\mathcal{N}$ are two countable models of RG. From above, we know that M and N are both countably infinite. List M as $m_0, m_1, m_2, \ldots$ and list N as $n_0, n_1, n_2, \ldots$. We build an isomorphism via a back-and-forth construction as in the proof of the corresponding result for DLO. In other words, we define a sequence of "partial isomorphisms" $h_k \colon M \to N$, i.e. each h_k will be a function from some finite subset of M to N that preserves the relation. More formally, we will have the following for each $k \in \mathbb{N}$:

- $\text{domain}(h_k)$ is a finite nonempty set.
- Each h_k is injective.
- For each $\ell \in \mathbb{N}$, we have $\{m_0, m_1, \ldots, m_\ell\} \subseteq \text{domain}(h_{2\ell})$.
- For each $\ell \in \mathbb{N}$, we have $\{n_0, n_1, \ldots, n_\ell\} \subseteq \text{range}(h_{2\ell+1})$.
- $h_k \subseteq h_{k+1}$, i.e. whenever $a \in \text{domain}(h_k)$, we have $a \in \text{domain}(h_{k+1})$ and $h_{k+1}(a) = h_k(a)$.
- Each h_k is a partial isomorphism, i.e. for all $a, b \in \text{domain}(h_k)$, we have $(a, b) \in \mathsf{R}^{\mathcal{M}}$ if and only if $(h_k(a), h_k(b)) \in \mathsf{R}^{\mathcal{N}}$.

We start by letting h_0 be the partial function with domain $\{m_0\}$ where $h_0(m_0) = n_0$, and then we let $h_1 = h_0$ (since n_0 is already in $\text{range}(h_0)$). Suppose that $k \in \mathbb{N}^+$ and we have defined h_k. We have two cases.

- *Case 1:* Suppose that k is odd, and fix $\ell \in \mathbb{N}$ with $k = 2\ell - 1$. If $m_\ell \in \text{domain}(h_k)$, let $h_{k+1} = h_k$. Suppose then that $m_\ell \notin \text{domain}(h_k)$. Let

$$\begin{aligned} A &= \{a \in \text{domain}(h_k) : (a, m_\ell) \in \mathsf{R}^{\mathcal{M}}\} \\ B &= \{b \in \text{domain}(h_k) : (b, m_\ell) \notin \mathsf{R}^{\mathcal{M}}\} \\ C &= \{h_k(a) : a \in A\} \\ D &= \{h_k(b) : b \in B\}. \end{aligned}$$

Since domain(h_k) is finite, we have that A and B are finite disjoint subsets of M with $A \cup B =$ domain(h_k). Since h_k is injective, we have that C and D are disjoint subsets of N with $C \cup D =$ range(h_k), and that $|A| = |C|$ and $|B| = |D|$. Now $\mathcal{N}$ is model of RG and $C \cap D = \emptyset$, so we can fix $w \in N \backslash (C \cup D)$ such that $(c, w) \in \mathsf{R}^{\mathcal{N}}$ for all $c \in C$ and $(d, w) \notin \mathsf{R}^{\mathcal{N}}$ for all $d \in D$. We now extend h_k to h_{k+1} by letting $h_{k+1}(m_\ell) = w$. It is straightforward to check that if h_k satisfies all of the above conditions, then h_{k+1} also satisfies all of the necessary conditions.

- *Case 2:* Suppose that k is even, and fix $\ell \in \mathbb{N}$ with $k = 2\ell$. If $n_\ell \in$ range(h_k), let $h_{k+1} = h_k$. Suppose then that $n_\ell \notin$ range(h_k). Let

$$C = \{c \in \text{range}(h_k) : (c, n_\ell) \in \mathsf{R}^{\mathcal{N}}\}$$
$$D = \{d \in \text{range}(h_k) : (d, n_\ell) \notin \mathsf{R}^{\mathcal{N}}\}$$
$$A = \{a \in \text{domain}(h_k) : h_k(a) \in C\}$$
$$B = \{b \in \text{domain}(h_k) : h_k(b) \in D\}.$$

Since domain(h_k) is finite, we have that range(h_k) is finite, and so C and D are finite disjoint subsets of N with $C \cup D =$ range(h_k). Since h_k is injective, we have that A and B are disjoint subsets of M with $A \cup B =$ domain(h_k), and that $|A| = |C|$ and $|B| = |D|$. Now $\mathcal{M}$ is model of RG and $A \cap B = \emptyset$, so we can fix $u \in M \backslash (A \cup B)$ such that $(a, u) \in \mathsf{R}^{\mathcal{M}}$ for all $a \in A$ and $(b, u) \notin \mathsf{R}^{\mathcal{M}}$ for all $b \in B$. We now extend h_k to h_{k+1} by letting $h_{k+1}(u) = n_\ell$. It is straightforward to check that if h_k satisfies all of the above conditions, then h_{k+1} also satisfies all of the necessary conditions.

Now define $h \colon M \to N$ by letting $h(m_\ell) = h_{2\ell}(m_\ell)$ for each $\ell \in \mathbb{N}$. Using the second through fifth conditions on the h_k, we conclude that h is a bijection. Now let $a, b \in M$ be arbitrary. Fix $k, \ell \in \mathbb{N}$ with $a = m_k$ and $b = m_\ell$. Let $t = \max\{k, \ell\}$. Since $a, b \in$ domain(h_{2t}), we have $(a, b) \in \mathsf{R}^{\mathcal{M}}$ if and only if $(h_{2t}(a), h_{2t}(b)) \in \mathsf{R}^{\mathcal{M}}$, which by fifth condition on the h_k is if and only if $(h(a), h(b)) \in \mathsf{R}^{\mathcal{M}}$. Therefore, h is an isomorphism. □

Corollary 6.4.9. *RG is a complete theory.*

Proof. Immediate from the Countable Łos-Vaught Test. □

Theorem 6.4.10. *Let $\tau \in Sent_{\mathcal{L}}$.*

1. *If $\tau \in RG$, then $\lim_{n\to\infty} Pr_n(\tau) = 1$.*
2. *If $\tau \notin RG$, then $\lim_{n\to\infty} Pr_n(\tau) = 0$.*

Proof.

1. Let $\tau \in RG$ be arbitrary. We then have $\Sigma \vDash \tau$, so by Compactness we may fix $N \in \mathbb{N}$ such that

$$\{\forall \mathsf{x} \neg \mathsf{Rxx}, \forall \mathsf{x} \forall \mathsf{y}(\mathsf{Rxy} \to \mathsf{Ryx})\} \cup \{\sigma_{r,s} : r, s \leq N\} \vDash \tau.$$

We then have that if $\mathcal{M} \in \mathcal{G}_n$ is such that $\mathcal{M} \vDash \neg\tau$, then

$$\mathcal{M} \vDash \bigvee_{0 \leq r,s \leq N, \max\{r,s\} > 0} \neg\sigma_{r,s}$$

Hence for every $n \in \mathbb{N}$ we have

$$Pr_n(\neg\tau) \leq \sum_{0 \leq r,s \leq N, \max\{r,s\} > 0} Pr_n(\neg\sigma_{r,s})$$

Since $\lim_{n\to\infty} Pr_n(\neg\sigma_{r,s}) = 0$ for each fixed choice of r and s, and since we have a finite sum, we conclude that $\lim_{n\to\infty} Pr_n(\neg\tau) = 0$. Therefore, $\lim_{n\to\infty} Pr_n(\tau) = 1$.

2. Suppose that $\tau \notin RG$. Since RG is complete, it follows that $\neg\tau \in RG$. Thus, $\lim_{n\to\infty} Pr_n(\neg\tau) = 1$ by part 1, and hence $\lim_{n\to\infty} Pr_n(\tau) = 0$.

□

With this background, the theorem that we want to prove is now immediate.

Proof of Theorem 6.4.3. Apply the previous theorem together with the fact that RG is complete. □

6.5 Exercises

1. Let $\mathcal{L} = \{\mathsf{R}\}$ where R is a binary relation symbol, and let $\mathsf{x}, \mathsf{y} \in Var$ be distinct.
   (a) Give a deduction showing that $\exists\mathsf{x}\forall\mathsf{y}\mathsf{Rxy} \vdash \forall\mathsf{y}\exists\mathsf{x}\mathsf{Rxy}$.
   (b) Show that $\forall\mathsf{y}\exists\mathsf{x}\mathsf{Rxy} \nvdash \exists\mathsf{x}\forall\mathsf{y}\mathsf{Rxy}$.

2. Let $\mathcal{L}$ be the basic group language, and let Σ be the group theory axioms, i.e.
$$\Sigma = \{\forall\mathsf{x}\forall\mathsf{y}\forall\mathsf{z}(\mathsf{fxfyz} = \mathsf{ffxyz}), \forall\mathsf{x}(\mathsf{fex} = \mathsf{x} \wedge \mathsf{fxe} = \mathsf{x}), \forall\mathsf{x}\exists\mathsf{y}(\mathsf{fxy} = \mathsf{e} \wedge \mathsf{fyx} = \mathsf{e})\}.$$
   Give a deduction showing that $\Sigma \vdash \forall\mathsf{x}(\mathsf{fex} = \mathsf{e} \to \mathsf{x} = \mathsf{e})$.

3. Let $\mathcal{L} = \{\mathsf{R}\}$ where R is a binary relation symbol.
   (a) Show that the class of directed acyclic graphs is not a strong elementary class in the language $\mathcal{L}$. (Exercise 1b in Chapter 4 asked you to show that it was an elementary class).
   (b) Show that the class of all connected graphs is not an elementary class in the language $\mathcal{L}$. (A graph (V, E) is *connected* if whenever $v, w \in V$ are distinct, there exists $u_1, u_2, \ldots, u_n \in V$ such that $u_1 = v$, $u_n = w$, and $(u_i, u_{i+1}) \in E$ for all i with $1 \leq i \leq n-1$).

4. Let $\mathcal{L} = \{\mathsf{e}, \mathsf{f}\}$ where e is a constant symbol and f is a binary function symbol.
   (a) Show that the class of all simple abelian groups is not a weak elementary class in the language $\mathcal{L}$. (Recall that an abelian group is simple if and only if it has no proper nontrivial subgroup, since all subgroups are normal.)
   (b) Show that the class of all torsion-free abelian groups (that is, abelian groups in which every nonidentity element has infinite order) is a weak elementary class but not an elementary class in the language $\mathcal{L}$

5. Let $\mathcal{L} = \{\mathsf{R}\}$ where R is a binary relation symbol. Let $\mathcal{Q}$ be the $\mathcal{L}$-structure $(\mathbb{Q}, <)$. Suppose that $\mathcal{M}$ is an infinite $\mathcal{L}$-structure which is a model of
$$\Sigma = \{\forall\mathsf{x}\neg\mathsf{Rxx}, \forall\mathsf{x}\forall\mathsf{y}\forall\mathsf{z}((\mathsf{Rxy} \wedge \mathsf{Ryz}) \to \mathsf{Rxz}), \forall\mathsf{x}\forall\mathsf{y}(\mathsf{Rxy} \vee \mathsf{Ryx} \vee \mathsf{x} = \mathsf{y})\}$$
   (i.e. $\mathcal{M}$ is an infinite strict linear ordering). Show that there exists a model $\mathcal{N}$ of Σ such that $\mathcal{M} \equiv \mathcal{N}$ and such that there exists an embedding from $\mathcal{Q}$ to $\mathcal{N}$.

   Stated more succinctly, show that for every infinite linear ordering, there exists an elementarily equivalent linear ordering that embeds the rationals.

Chapter 7

Model Theory

In many mathematical areas (like partial orderings, groups, or graphs), we write down some axioms and immediately have several different models of those axioms in mind. In the setting of first-order logic, this corresponds to writing down a set Σ of sentences in a language, and looking at the elementary class $Mod(\Sigma)$. Since $Mod(\Sigma) = Mod(Cn(\Sigma))$ by Proposition 5.2.3, and $Cn(\Sigma)$ is a theory by Proposition 5.2.4, we can view this situation as looking at the (elementary) class of models of a theory.

If we create a theory T in a different way (rather than as $Cn(\Sigma)$ for some easily written down sentences Σ), then there are situations where we might only envision one model of T. For example, consider the language $\mathcal{L} = \{\mathsf{0}, \mathsf{1}, +, \cdot, <\}$ of arithmetic, where $\mathsf{0}, \mathsf{1}$ are constant symbols, $<$ is a binary relation symbol, and $+, \cdot$ are binary function symbols. Let $\mathcal{M} = (\mathbb{N}, 0, 1, +, \cdot, <)$ where the symbol $\mathsf{0}$ is interpreted as the real 0, the symbol $+$ is interpreted as the real addition, etc. Does $Th(\mathcal{M})$ completely determine the model $\mathcal{M}$? In other words, is every model of $Th(\mathcal{M})$ isomorphic to $\mathcal{M}$?

We have already answered this question in great generality. Using (the note after) Exercise 7 in Chapter 4, we know that if $\mathcal{M}$ is an structure of a finite language $\mathcal{L}$, then any model of $Th(\mathcal{M})$ is isomorphic to $\mathcal{M}$. However, If T is a theory in a countable language that has at least one infinite model, then we know from Theorem 5.1.9 and Proposition 6.3.4 that T has both a countable model and an uncountable model. In particular, if $\mathcal{M}$ is an infinite structure in a countable language, then $Th(\mathcal{M})$ will have at least two models that are not isomorphic. Once we have developed the set-theoretic tools, we will eventually show that if T is a theory in language $\mathcal{L}$ that has at least one infinite model, then it has a model of every infinite cardinality greater than or equal to the cardinality of $\mathcal{L}$. Thus, outside of the realm of the finite, first-order logic is not powerful enough to put any control on the existence of models of various infinite sizes.

In particular, there are many models of $Th((\mathbb{N}, 0, 1, +, \cdot, <))$ that are not isomorphic to the natural numbers! We will spend some time examining the structure of such models in Section 7.2. More generally, we will make extensive use of the Compactness Theorem in this chapter to begin a deeper exploration of the class of models of a given theory.

7.1 Diagrams and Embeddings

Given an $\mathcal{L}$-structure $\mathcal{M}$, the set $Th(\mathcal{M}) = \{\sigma \in Sent_{\mathcal{L}} : \mathcal{M} \vDash \sigma\}$ contains all of the first-order facts that are true in $\mathcal{M}$. As we have seen, the amount of information contained in this set can vary based on the language and the structure. For example, we know that $Th((\mathbb{Q}, <)) = Th((\mathbb{R}, <))$, and that this common theory is just DLO, which is the set of consequences of a few simple axioms. In contrast, if we work in the language of rings, then

$$Th((\mathbb{Q}, \mathsf{0}, \mathsf{1}, +, -, \cdot)) \neq Th((\mathbb{R}, \mathsf{0}, \mathsf{1}, +, -, \cdot))$$

because the latter contains $\exists \mathsf{x}(\mathsf{x} \cdot \mathsf{x} = \mathsf{1} + \mathsf{1})$ but the former does not does not.

Let's explore the theory of the structure $\mathcal{M} = (\mathbb{N}, 0, 1, +, \cdot, <)$ in the language $\mathcal{L}$ of arithmetic. As mentioned above, there are models of $Th(\mathcal{M})$ that are not isomorphic to $\mathcal{M}$. To begin to understand such models, we first investigate a few peculiarities of the structure $\mathcal{M}$. Recall from Definition 4.6.14 that a closed term is a term that does not contain any variables. We can obtain some closed terms of $\mathcal{L}$ by simply adding the constant symbol $\mathsf{1}$ to itself many times. Since every nonzero element of $M = \mathbb{N}$ can be obtained by adding 1 to itself a finite number of times, it follows that every element of M can be "named" by a closed term. Here is the formal definition.

Definition 7.1.1. *Let $\mathcal{L}$ be the language of arithmetic. For each $n \in \mathbb{N}$, we define a term $\underline{n} \in Term_{\mathcal{L}}$ as follows. Let $\underline{0} = \mathsf{0}$ and let $\underline{1} = \mathsf{1}$. Now define $\underline{n}$ recursively by letting $\underline{n+1} = \underline{n} + \mathsf{1}$ for each $n \geq 1$.*

Notice here that the 1 and the $+$ in $\underline{n+1}$ means the actual number 1 and the actual addition function, whereas the $\mathsf{1}$ and $+$ in $\underline{n} + \mathsf{1}$ mean the symbols $\mathsf{1}$ and $+$ in our language $\mathcal{L}$. Thus, for example, $\underline{2}$ is the term $\mathsf{1} + \mathsf{1}$ and $\underline{3}$ is the term $(\mathsf{1} + \mathsf{1}) + \mathsf{1}$. Each $\underline{n}$ is a closed term, so given an $\mathcal{L}$-structure $\mathcal{M}$ and an $n \in \mathbb{N}$, recall that we write $\underline{n}^{\mathcal{M}}$ to mean $\overline{s}(t)$ for some (any) variable assignment $s\colon Var \to M$. In the structure $\mathcal{M} = (\mathbb{N}, 0, 1, +, \cdot, <)$, we have that $\underline{n}^{\mathcal{M}} = n$ for all $n \in \mathbb{N}$ by a trivial induction.

Using these special closed terms, we can write down simple sentences like $\underline{2} + \underline{3} = \underline{5}$ that are true in $\mathcal{M}$ and hence are elements of $Th(\mathcal{M})$. In fact, for any $m, n \in \mathbb{N}$, the sentences $\underline{m} + \underline{n} = \underline{m+n}$ and $\underline{m} \cdot \underline{n} = \underline{m \cdot n}$ are elements of $Th(\mathcal{M})$. Similarly, we can write down some important sentences about $<$. For example, $\underline{3} < \underline{7}$ is in $Th(\mathcal{M})$, as is the more complicated

$$\forall \mathsf{x}(\mathsf{x} < \underline{3} \to (\mathsf{x} = \underline{0} \vee \mathsf{x} = \underline{1} \vee \mathsf{x} = \underline{2})).$$

Any model $Th(\mathcal{M})$ must satisfy all of these sentences. Notice that the sentences like $\underline{m} + \underline{n} = \underline{m+n}$ are just basic atomic sentences. Now if $\mathcal{N}$ is a model of these literals (see Definition 5.4.5), then the set $\{\underline{n}^{\mathcal{N}} : n \in \mathbb{N}\}$ is a part of the model $\mathcal{N}$ where addition in $\mathcal{N}$ behaves just like addition in the natural numbers. In fact, thinking about the corresponding literals for the other function and relation symbols, it follows that $\{\underline{n}^{\mathcal{N}} : n \in \mathbb{N}\}$ is a substructure of $\mathcal{N}$ that is isomorphic to $\mathcal{M}$. We make this precise with the following general result. Notice that part (2) implies that $\{\underline{n}^{\mathcal{N}} : n \in \mathbb{N}\}$ is actually an *elementary substructure* of $\mathcal{N}$ that is isomorphic to $\mathcal{M}$, but it uses the fact that $\mathcal{N}$ satisfies more than just the literals.

Proposition 7.1.2. *Let $\mathcal{M}$ be an $\mathcal{L}$-structure. Assume that for every $a \in M$, there exists a closed term $t \in Term_{\mathcal{L}}$ such that $t^{\mathcal{M}} = a$.*

1. *If $\mathcal{N}$ is a model of $\{\sigma \in Sent_{\mathcal{L}} : \sigma \in Literal_{\mathcal{L}} \text{ and } \mathcal{M} \vDash \sigma\}$, then the function $h\colon M \to N$ defined by letting $h(a) = t^{\mathcal{N}}$, where $t \in Term_{\mathcal{L}}$ is some closed term with $t^{\mathcal{M}} = a$, is a well-defined embedding of $\mathcal{M}$ into $\mathcal{N}$.*

2. *If $\mathcal{N}$ is a model of $Th(\mathcal{M}) = \{\sigma \in Sent_{\mathcal{L}} : \mathcal{M} \vDash \sigma\}$, then the function h described above is an elementary embedding of $\mathcal{M}$ into $\mathcal{N}$.*

Proof. Assume first that $\mathcal{N}$ is a model of $\{\sigma \in Sent_{\mathcal{L}} : \sigma \in Literal_{\mathcal{L}} \text{ and } \mathcal{M} \vDash \sigma\}$. We check that h is a well-defined embedding.

- h is well-defined: Let $a \in M$ be arbitrary. Suppose that $t, u \in Term_{\mathcal{L}}$ are both closed terms with $t^{\mathcal{M}} = a = u^{\mathcal{M}}$. We then have that $\mathcal{M} \vDash t = u$, so since $t = u$ is a sentence and a literal, it follows that $\mathcal{N} \vDash t = u$, and hence $t^{\mathcal{N}} = u^{\mathcal{N}}$. Therefore, h is well-defined.

- h is injective: Let $a, b \in M$ be arbitrary with $a \neq b$. By assumption, we can fix closed terms $t, u \in Term_{\mathcal{L}}$ with $t^{\mathcal{M}} = a$ and $u^{\mathcal{M}} = b$. We then have that $\mathcal{M} \vDash \neg(t = u)$, so since $\neg(t = u)$ is a sentence and a literal, it follows that $\mathcal{N} \vDash \neg(t = u)$, so $t^{\mathcal{N}} \neq u^{\mathcal{N}}$, and hence $h(a) \neq h(b)$. Therefore, h is injective.

- h preserves constants: Let $\mathsf{c} \in \mathcal{C}$ be a constant symbol. Since c is a closed term, we have $h(\mathsf{c}^{\mathcal{M}}) = \mathsf{c}^{\mathcal{N}}$ immediately by definition.

- h preserves relations: Let $\mathsf{R} \in \mathcal{R}_k$ be a relation symbol of $\mathcal{L}$, and let $a_1, a_2, \ldots, a_k \in M$ be arbitrary. For each i, fix a closed term $t_i \in Term_{\mathcal{L}}$ with $t_i^{\mathcal{M}} = a_i$. We have two cases:

  - *Case 1:* Suppose that $(a_1, a_2, \ldots, a_k) \in \mathsf{R}^{\mathcal{M}}$. We then have that $\mathcal{M} \vDash \mathsf{R}t_1t_2\cdots t_k$, so since $\mathsf{R}t_1t_2\cdots t_k$ is a sentence and a literal, it follows that $\mathcal{N} \vDash \mathsf{R}t_1t_2\cdots t_k$. Therefore, we have $(t_1^{\mathcal{N}}, t_2^{\mathcal{N}}, \ldots, t_k^{\mathcal{N}}) \in \mathsf{R}^{\mathcal{N}}$, and hence $(h(a_1), h(a_2), \ldots, h(a_k)) \in \mathsf{R}^{\mathcal{N}}$.
  - *Case 2:* Suppose that $(a_1, a_2, \ldots, a_k) \notin \mathsf{R}^{\mathcal{M}}$. We then have that $\mathcal{M} \vDash \neg\mathsf{R}t_1t_2\cdots t_k$, so since $\neg\mathsf{R}t_1t_2\cdots t_k$ is a sentence and a literal, it follows that $\mathcal{N} \vDash \neg\mathsf{R}t_1t_2\cdots t_k$. Therefore, we have $(t_1^{\mathcal{N}}, t_2^{\mathcal{N}}, \ldots, t_k^{\mathcal{N}}) \notin \mathsf{R}^{\mathcal{N}}$, and hence $(h(a_1), h(a_2), \ldots, h(a_k)) \notin \mathsf{R}^{\mathcal{N}}$.

  Thus, $(a_1, a_2, \ldots, a_k) \in \mathsf{R}^{\mathcal{M}}$ if and only if $(h(a_1), h(a_2), \ldots, h(a_k)) \in \mathsf{R}^{\mathcal{N}}$.

- h preserves functions: Let $\mathsf{f} \in \mathcal{F}_k$ be a function symbol of $\mathcal{L}$, and let $a_1, a_2, \ldots, a_k \in M$ be arbitrary. Let
$$b = \mathsf{f}^{\mathcal{M}}(a_1, a_2, \ldots, a_k).$$
For each i, fix a closed term $t_i \in Term_{\mathcal{L}}$ with $t_i^{\mathcal{M}} = a_i$, and also fix a closed term $u \in Term_{\mathcal{L}}$ with $u^{\mathcal{M}} = b$. We then have that $\mathcal{M} \vDash \mathsf{f}t_1t_2\cdots t_k = u$, so since $\mathsf{f}t_1t_2\cdots t_k = u$ is a sentence and a literal, it follows that $\mathcal{N} \vDash \mathsf{f}t_1t_2\cdots t_k = u$. Therefore, we have $\mathsf{f}^{\mathcal{N}}(t_1^{\mathcal{N}}, t_2^{\mathcal{N}}, \ldots, t_k^{\mathcal{N}}) = u^{\mathcal{N}}$, and hence $\mathsf{f}^{\mathcal{N}}(h(a_1), h(a_2), \ldots, h(a_k)) = h(b)$. Thus,
$$h(\mathsf{f}^{\mathcal{M}}(a_1, a_2, \ldots, a_k)) = h(b) = \mathsf{f}^{\mathcal{N}}(h(a_1), h(a_2), \ldots, h(a_k)).$$

Therefore, under the assumptions of (1), we have shown that h is a well-defined embedding of $\mathcal{M}$ into $\mathcal{N}$.

Suppose now that $\mathcal{N}$ is a model of $\{\sigma \in Sent_{\mathcal{L}} : \mathcal{M} \vDash \sigma\}$. We need to show that h is an elementary embedding. Let $\varphi(\mathsf{x}_1, \mathsf{x}_2, \ldots, \mathsf{x}_k) \in Form_{\mathcal{L}}$ and let $a_1, a_2, \ldots, a_k \in M$ be arbitrary. Assume first that
$$(\mathcal{M}, a_1, a_2, \ldots, a_k) \vDash \varphi.$$
For each i, fix a closed term $t_i \in Term_{\mathcal{L}}$ with $t_i^{\mathcal{M}} = a_i$. Let σ be the sentence obtained by simultaneously substituting the t_i for the x_i. By Corollary 4.6.15, we have that $\mathcal{M} \vDash \sigma$. By assumption, it follows that $\mathcal{N} \vDash \sigma$. Using Corollary 4.6.15 again, together with the fact that $t_i^{\mathcal{N}} = h(a_i)$ for all i, we conclude that
$$(\mathcal{N}, h(a_1), h(a_2), \ldots, h(a_k)) \vDash \varphi.$$
Conversely, if $(\mathcal{M}, a_1, a_2, \ldots, a_k) \nvDash \varphi$, then $(\mathcal{M}, a_1, a_2, \ldots, a_k) \vDash \neg\varphi$, so the above argument shows that $(\mathcal{N}, h(a_1), h(a_2), \ldots, h(a_k)) \vDash \neg\varphi$, from which we conclude that $(\mathcal{N}, h(a_1), h(a_2), \ldots, h(a_k)) \nvDash \varphi$. Therefore, h is an elementary embedding. □

As mentioned before the proposition, since $\mathcal{M} = (\mathbb{N}, 0, 1, +, \cdot, <)$, it follows that given any model $\mathcal{N}$ of $Th(\mathcal{M})$, we can find an elementary embedding of $\mathcal{M}$ into $\mathcal{N}$. Thus, any model of $Th(\mathcal{M})$ contains an elementary substructure that is isomorphic to $\mathcal{M}$. Alternatively, if we identify $\mathcal{M}$ with this elementary substructure, then every model $\mathcal{N}$ of $Th(\mathcal{M})$ can be viewed as an elementary extension of $\mathcal{M}$. We will explore these models of $Th(\mathcal{M})$ in more detail in Section 7.2, but we now spend the rest of section thinking about how to cope with the assumption of Proposition 7.1.2.

Of course, there are many structures $\mathcal{M}$ where some elements are *not* named by closed terms. In the language of group theory (which does have a constant symbol e), if $\mathcal{M}$ is an $\mathcal{L}$-structure that is a group, then for every closed term t, we have $t^{\mathcal{M}} = e$. Thus, the only groups in the language $\mathcal{L}$ in which every element is named by a closed term are the trivial groups. In a language without constant symbols, such as the language

$\mathcal{L} = \{\mathsf{R}\}$ where R is a binary relation symbol (suitable for partial orderings, equivalence relations, graphs, etc.), then there are no closed terms, so $\mathcal{L}$-structures never have this property!

For another example, consider the language $\mathcal{L}$ of ordered rings, and let $\mathcal{R} = (\mathbb{R}, \mathsf{0}, \mathsf{1}, +, -, \cdot, <)$. Now it is straightforward to see that every element of $\mathbb{Z}$ is named by a closed term, but no elements of $\mathbb{R}\backslash\mathbb{Z}$ have this property. In fact, the conclusion of Proposition 7.1.2 is false! There are models of $Th(\mathcal{R})$ that do not embed $\mathcal{R}$ (let alone elementarily embed), because there is a countable model of $Th(\mathcal{R})$ by the Countable Lowenheim-Skolem Theorem.

Although these examples seem to suggest that Proposition 7.1.2 is of extremely limited usefulness, we can always change the language. If we have a given $\mathcal{L}$-structure $\mathcal{M}$ in hand, then we can add a (potentially large) collection of constant symbols to the language $\mathcal{L}$ so that every element of M has such a symbol associated to it.

Definition 7.1.3. *Let $\mathcal{M}$ be an $\mathcal{L}$-structure.*

- *For each $a \in M$, let $\underline{a}$ be a new constant symbol that does not appear in the language $\mathcal{L}$.*
- *Let $\mathcal{L}_M$ be the language $\mathcal{L}$ together with the addition of $\{\underline{a} : a \in M\}$ to the set of constant symbols.*
- *Let $\mathcal{M}_{exp}$ be the $\mathcal{L}_M$-structure that is obtained by expanding (see Definition 4.3.13) $\mathcal{M}$ by letting $\underline{a}^{\mathcal{M}_{exp}} = a$ for all $a \in M$.*
- *Let $Diag(\mathcal{M}) = \{\sigma \in Sent_{\mathcal{L}_M} : \sigma \in Literal_{\mathcal{L}_M} \text{ and } \mathcal{M}_{exp} \vDash \sigma\}$.*
- *Let $Diag_{el}(\mathcal{M}) = \{\sigma \in Sent_{\mathcal{L}_\mathcal{M}} : \mathcal{M}_{exp} \vDash \sigma\}$.*

We call $Diag(\mathcal{M})$ the (atomic) diagram *of $\mathcal{M}$, and call $Diag_{el}(\mathcal{M})$ the* elementary diagram *of $\mathcal{M}$.*

Suppose that we are working in the group theory language $\mathcal{L}$, and we have an $\mathcal{L}$-structure $\mathcal{M}$ that is a group. The elements of $Diag(\mathcal{M})$ express very simple facts about whether certain elements of M are equal or are not equal. For example, the sentences $\underline{a} * \underline{c} = \underline{e}$ and $\neg(\underline{a} * \underline{a} = \underline{a})$ are what typical elements of $Diag(\mathcal{M})$ look like. Note that there are two distinct constant symbols e and $\underline{e}$ that are both interpreted as the identity of M, so there is some redundancy. We also have literals like $\neg((\underline{a} * \mathsf{e}) \cdot \underline{c} = \underline{b} * \underline{a})$. However, the basic sentences of the form $\underline{a} * \underline{b} = \underline{c}$ provide all of the information that appears in the Cayley table (i.e. multiplication table) of the group. In fact, it often useful to think of $Diag(\mathcal{M})$ as generalizing the idea of a Cayley table to any structure.

Now for any $\mathcal{L}$-structure $\mathcal{M}$, the expanded $\mathcal{L}_M$-structure $\mathcal{M}_{exp}$ has the property that every element of the universe is named by a closed term, which leads to the following consequence.

Corollary 7.1.4. *Let $\mathcal{L}$ be a language and let $\mathcal{M}$ be an $\mathcal{L}$-structure. Let $\mathcal{N}$ be an $\mathcal{L}_M$-structure, and define $h\colon M \to N$ by letting $h(a) = \underline{a}^{\mathcal{N}}$.*

1. *If $\mathcal{N}$ is a model of $Diag(\mathcal{M})$, then h is an embedding from $\mathcal{M}_{exp}$ to $\mathcal{N}$, and hence h is an embedding from $\mathcal{M}$ to $\mathcal{N} \restriction \mathcal{L}$.*
2. *If $\mathcal{N}$ is a model of $Diag_{el}(\mathcal{M})$, then h is an elementary embedding from $\mathcal{M}_{exp}$ to $\mathcal{N}$, and hence h is an elementary embedding from $\mathcal{M}$ to $\mathcal{N} \restriction \mathcal{L}$.*

Proof. Notice that for every $a \in M$, there trivially exists a closed term $t \in Term_{\mathcal{L}_M}$ such that $t^{\mathcal{M}_{exp}} = a$, because we can just take $t = \underline{a}$.

Suppose now $\mathcal{N}$ is a model of $Diag(\mathcal{M})$. Proposition 7.1.2 immediately implies that h is an embedding from $\mathcal{M}_{exp}$ to $\mathcal{N}$. Since $\mathcal{M}$ and $\mathcal{N} \restriction \mathcal{L}$ are simply the restrictions of $\mathcal{M}_{exp}$ and $\mathcal{N}$ to the language $\mathcal{L}$, we immediately conclude that h is an embedding of these restricted structures as well. If in addition $\mathcal{N}$ is a model of $Diag_{el}(\mathcal{M})$, then part (2) of Proposition 7.1.2 implies that h is an elementary embedding from $\mathcal{M}_{exp}$ to $\mathcal{N}$, and hence of the restricted structures as well. □

One way to use Corollary 7.1.4 is as a tool to determine whether an $\mathcal{L}$-structure $\mathcal{N}$ embeds another $\mathcal{L}$-structure $\mathcal{M}$. If we can expand $\mathcal{N}$ to interpret the new constants $\underline{a}$ of $\mathcal{L}_M$ in such a way that the resulting expanded structure is a model of $Diag(\mathcal{M})$ (i.e. if we can find elements of N that respect the generalized Cayley table of $\mathcal{M}$), then we can indeed embed $\mathcal{M}$. Similarly, if the resulting structure is a model of all of the sentences in $Diag_{el}(\mathcal{M})$, then we can find an elementary embedding of $\mathcal{M}$.

However, the real power of the result comes from combining it with the Compactness Theorem to *build* models of $Th(\mathcal{M})$ that embed $\mathcal{M}$. Our first result in this direction says that we can always extend a structure $\mathcal{M}$ to a strictly larger world where $\mathcal{M}$ sits inside as an elementary substructure.

Corollary 7.1.5. *Let $\mathcal{L}$ be a language. For every infinite $\mathcal{L}$-structure $\mathcal{M}$, there exists an $\mathcal{L}$-structure $\mathcal{N}$ with $\mathcal{M} \preceq \mathcal{N}$ and $M \subsetneq N$. Moreover, if $\mathcal{L}$ and $\mathcal{M}$ are both countable, then there exists a countable such $\mathcal{N}$.*

Proof. Let $\mathcal{L}' = \mathcal{L}_M \cup \{\mathsf{c}\}$, where c is a new constant symbol. Let

$$\Sigma = Diag_{el}(\mathcal{M}) \cup \{\neg(\mathsf{c} = \underline{a}) : a \in M\}.$$

Since M is infinite, every finite subset of Σ is satisfiable (by taking $\mathcal{M}_{exp}$ and then interpreting c as an element of M distinct from each $a \in M$ whose name appears in the finite set). Therefore, Σ is satisfiable by Compactness. Fix a model $\mathcal{N}$ of Σ. Since $\mathcal{N} \upharpoonright \mathcal{L}_M$ is an $\mathcal{L}_M$ structure that is a model $Diag_{el}(\mathcal{M})$, it follows from Corollary 7.1.4 that the function $h\colon M \to N$ defined by $h(a) = \underline{a}^{\mathcal{N}}$ is an elementary embedding from $\mathcal{M}$ to $\mathcal{N} \upharpoonright \mathcal{L}$. Notice also that $\mathsf{c}^{\mathcal{N}} \neq \underline{a}^{\mathcal{N}}$ for each $a \in M$, so $\mathsf{c}^{\mathcal{N}} \notin \text{range}(h)$. By identifying $\mathcal{M}$ with its image (i.e. by renaming each of the elements of $\text{range}(h) \subseteq N$ to be the corresponding input from M that hits it), we can view $\mathcal{N}$ as an elementary extension of $\mathcal{M}$.

For the last statement, notice that if $\mathcal{L}$ and $\mathcal{M}$ are both countable, then $\mathcal{L}_M \cup \{\mathsf{c}\}$ is countable, so there exists a countable model $\mathcal{N}$ of Σ by the Countable Lowenheim-Skolem Theorem. □

Suppose that $\mathcal{L}$ is a countable language. Recall that the Countable Lowenheim-Skolem-Tarski Theorem (see Theorem 4.5.4) says then every $\mathcal{L}$-structure had a countable elementary substructure. In particular, every uncountable structure has a proper elementary substructure. In contrast, we just showed that every infinite structure can be strictly extended to an elementary superstructure. Moreover, we can follow the idea of the proof of Proposition 6.3.4 to show that we can extend every structure to an *uncountable* elementary superstructure.

Corollary 7.1.6. *Let $\mathcal{L}$ be a language. For every infinite $\mathcal{L}$-structure $\mathcal{M}$, there exists an uncountable $\mathcal{L}$-structure $\mathcal{N}$ with $\mathcal{M} \preceq \mathcal{N}$.*

Proof. Let $\mathcal{L}' = \mathcal{L}_M \cup \{\mathsf{c}_a : a \in \mathbb{R}\}$ where the c_a are new distinct constant symbols, and let

$$\Sigma = Diag_{el}(\mathcal{M}) \cup \{\neg(\mathsf{c}_a = \mathsf{c}_b) : a, b \in \mathbb{R} \text{ and } a \neq b\}.$$

Using the fact that M is finite, every finite subset of Σ has a model by interpreting the c_a that appear differently (see the proof of Proposition 6.3.4). Therefore, Σ has a model $\mathcal{N}$ by Compactness, which is necessarily uncountable. Now follow the proof of the previous corollary to argue that there is an elementary embedding of $\mathcal{M}$ to $\mathcal{N} \upharpoonright \mathcal{L}$, and hence we can view $\mathcal{M}$ as an elementary substructure of $\mathcal{N} \upharpoonright \mathcal{L}$. □

Before moving on, we prove the converse of Corollary 7.1.4. That is, instead of starting with a model $\mathcal{N}$ of $Diag(\mathcal{M})$ and showing that the natural map from M to N is an embedding, we instead show that if the natural map is an embedding (resp. elementary embedding), then the corresponding expansion of $\mathcal{N}$ is a model of $Diag(\mathcal{M})$ (resp. $Diag_{el}(\mathcal{M})$).

Proposition 7.1.7. *Let $\mathcal{M}$ and $\mathcal{N}$ be $\mathcal{L}$-structures, and let $h\colon M \to N$ be a function. Expand $\mathcal{N}$ to an $\mathcal{L}_M$-structure $\mathcal{N}'$ by letting $\underline{a}^{\mathcal{N}'} = h(a)$ for all $a \in M$.*

1. *If h is an embedding of $\mathcal{M}$ into $\mathcal{N}$, then $\mathcal{N}'$ is a model of $Diag(\mathcal{N})$.*

2. If h is an elementary embedding of $\mathcal{M}$ into $\mathcal{N}$, then $\mathcal{N}'$ is a model of $Diag_{el}(\mathcal{N})$.

The proof of part (1) is reasonably straightforward (see below) with the tools that we have developed. However, part (2) requires a little thought. Suppose that we want to prove that $\mathcal{N}'$ is a model of $Diag_{el}(\mathcal{M})$. The obstacle is that sentences in $Diag_{el}(\mathcal{M})$ might not be elements of $Sent_{\mathcal{L}}$ as they might contain constant symbols from $\mathcal{L}_M \backslash \mathcal{L}$, which is an issue because the assumption that h is an elementary embedding only gives us information about formulas over $\mathcal{L}$. However, if the new constant symbols appearing in a sentence $\sigma \in Diag_{el}(\mathcal{M})$ are $\underline{a_1}, \underline{a_2}, \ldots, \underline{a_n}$, then it is intuitively clear that there exists $\psi(\mathsf{y}_1, \ldots, \mathsf{y}_n) \in Form_{\mathcal{L}}$ such that σ is the result of simultaneously substituting the $\underline{a_i}$ for the variables y_i. The following result states this fact in a general form. Although we originally defined simultaneous substitution of terms for variables, we can also define such simultaneous substitution of terms for constant symbols (we leave the formal recursive definition to the reader).

Proposition 7.1.8. *Let $\mathcal{L} \subseteq \mathcal{L}'$ be languages such that the only new symbols of $\mathcal{L}'$ are constant symbols. Let $\mathcal{M}$ be an $\mathcal{L}'$-structure, and let $\varphi(\mathsf{x}_1, \ldots, \mathsf{x}_k) \in Form_{\mathcal{L}'}$. Suppose that the new constants (i.e. the elements of $\mathcal{L}'\backslash\mathcal{L}$) that occur in φ are $\mathsf{c}_1, \ldots, \mathsf{c}_n$. Let $\mathsf{y}_1, \ldots, \mathsf{y}_n$ be variables that do not occur in φ, and let ψ be the result of simultaneously substituting the y_i for the constants c_i. We then have that $\psi(\mathsf{x}_1, \ldots, \mathsf{x}_k, \mathsf{y}_1, \ldots, \mathsf{y}_n) \in Form_{\mathcal{L}}$ and that φ is the result of simultaneously substituting the constants c_i for the y_i. Moreover, for any $a_1, a_2, \ldots, a_k \in M$, we have*

$$(\mathcal{M}, a_1, \ldots, a_k) \vDash \varphi \text{ if and only if } (\mathcal{M}, a_1, \ldots, a_k, \mathsf{c}_1^{\mathcal{M}}, \ldots, \mathsf{c}_n^{\mathcal{M}}) \vDash \psi.$$

Proof. Let $\alpha \colon Var \to Term_{\mathcal{L}_M}$ be the function that codes simultaneous substitution of the (closed terms) c_i for the y_i, while leaving other variables alone, as described in Corollary 4.6.15. Similarly, let $\beta \colon \mathcal{C} \to Term_{\mathcal{L}}$ be the function that codes simultaneous substitution of the y_i for the c_i, while leaving all other constants alone. Notice that $\psi = \varphi^{\beta}$ by definition. It is then tedious but straightforward to check that $\psi \in Form_{\mathcal{L}}$ and (since the y_i do not occur in φ) that $\varphi = (\varphi^{\beta})^{\alpha}$, so $\varphi = \psi^{\alpha}$.

Now let $a_1, \ldots, a_k \in M$ be arbitrary. Using Corollary 4.6.15 together with the fact that $\psi^{\alpha} = \varphi$, it follows that

$$(\mathcal{M}, a_1, \ldots, a_k, \mathsf{c}_1^{\mathcal{M}}, \ldots, \mathsf{c}_n^{\mathcal{M}}) \vDash \psi \text{ if and only if } (\mathcal{M}, a_1, \ldots, a_k) \vDash \varphi,$$

completing the proof. □

We now return to the proof of Proposition 7.1.7.

Proof of Proposition 7.1.7.

1. Suppose that h is an embedding of $\mathcal{L}$-structures from $\mathcal{M}$ to $\mathcal{N}$. We first show that h is an embedding (of $\mathcal{L}_M$-structures) from $\mathcal{M}_{exp}$ to $\mathcal{N}'$. Since $\mathcal{M}_{exp}$ and $\mathcal{N}'$ are expansions of these structures to the language $\mathcal{L}_M$, we need only check that h preserves the new constant symbols. Now for any $a \in M$, we have
$$h(\underline{a}^{\mathcal{M}_{exp}}) = h(a) = \underline{a}^{\mathcal{N}'}.$$
Thus, h is indeed an embedding from $\mathcal{M}_{exp}$ to $\mathcal{N}'$. Applying Theorem 4.3.6, we know that for any quantifier-free $\varphi \in Form_{\mathcal{L}_M}$ and any $s \colon Var \to M$, we have
$$(\mathcal{M}_{exp}, s) \vDash \varphi \Leftrightarrow (\mathcal{N}', h \circ s) \vDash \varphi.$$
Thus, if $\sigma \in Sent_{\mathcal{L}_M}$ is quantifier-free, then since σ has no free variables, it follows that
$$\mathcal{M}_{exp} \vDash \sigma \Leftrightarrow \mathcal{N}' \vDash \sigma.$$
In particular, since every element of $Diag(\mathcal{M})$ is a quantifier-free sentence, we have $\mathcal{N}' \vDash \sigma$ for all $\sigma \in Diag(\mathcal{M})$. Therefore, $\mathcal{N}'$ is a model of $Diag(\mathcal{M})$.

2. Suppose that h is an elementary embedding of $\mathcal{L}$-structures from $\mathcal{M}$ to $\mathcal{N}$. By definition, we know that for every $\varphi(\mathsf{x}_1, \mathsf{x}_2, \ldots, \mathsf{x}_k) \in Form_{\mathcal{L}}$ and every $b_1, b_2, \ldots, b_k \in M$, we have

$$(\mathcal{M}, b_1, b_2, \ldots, b_n) \vDash \varphi \text{ if and only if } (\mathcal{N}, h(b_1), h(b_2), \ldots, h(b_n)) \vDash \varphi.$$

Now let $\sigma \in Diag_{el}(\mathcal{M})$ be arbitrary. Suppose that the new constants (i.e. the elements of $\mathcal{L}_M \backslash \mathcal{L}$) that appear in σ are $\underline{b_1}, \ldots, \underline{b_n}$. Let $\mathsf{y}_1, \ldots, \mathsf{y}_n$ be new variables that do not occur in σ, and let $\psi(\mathsf{y}_1, \ldots, \mathsf{y}_n)$ be the result of simultaneously substituting the variables y_i for the corresponding $\underline{b_i}$. Using Proposition 7.1.8 applied to the languages $\mathcal{L} \subseteq \mathcal{L}_M$ and the $\mathcal{L}_M$-structure $\mathcal{M}_{exp}$, we have $\psi(\mathsf{y}_1, \ldots, \mathsf{y}_k) \in Form_{\mathcal{L}}$ and

$$\mathcal{M}_{exp} \vDash \sigma \text{ if and only if } (\mathcal{M}_{exp}, \underline{b_1}^{\mathcal{M}_{exp}}, \ldots, \underline{b_n}^{\mathcal{M}_{exp}}) \vDash \psi.$$

Since $\sigma \in Diag_{el}(\mathcal{M})$, we know that $\mathcal{M}_{exp} \vDash \sigma$. Since $\underline{b_i}^{\mathcal{M}_{exp}} = b_i$ for all i, and since $\psi \in Form_{\mathcal{L}}$, we can use Proposition 4.3.14 to conclude that $(\mathcal{M}, b_1, \ldots, b_n) \vDash \psi$. Since h is an elementary embedding, it follows that $(\mathcal{N}, h(b_1), \ldots, h(b_n)) \vDash \psi$, so $(\mathcal{N}, \underline{b_1}^{\mathcal{N}'}, \ldots, \underline{b_n}^{\mathcal{N}'}) \vDash \psi$, and hence $(\mathcal{N}', b_1^{\mathcal{N}'}, \ldots, b_n^{\mathcal{N}'}) \vDash \psi$. Using Proposition 7.1.8 again, but now applied to the $\mathcal{L}_M$-structure $\mathcal{N}'$, we conclude that $\mathcal{N}' \vDash \sigma$. Since $\sigma \in Diag_{el}(\mathcal{M})$ was arbitrary, it follows that $\mathcal{N}'$ is a model of $Diag_{el}(\mathcal{M})$.

□

Finally, we can use diagrams together with the ideas in the previous proofs to give another, more natural, interpretation of the concept of definability with parameters (see Definition 4.4.6).

Corollary 7.1.9. *Let $\mathcal{M}$ be an $\mathcal{L}$-structure. Suppose that $k \in \mathbb{N}^+$ and $X \subseteq M^k$. We then have that X is definable with parameters in $\mathcal{M}$ if and only if X is definable in $\mathcal{M}_{exp}$.*

Proof. Assume first that X is definable with parameters in $\mathcal{M}$. Fix $\varphi(\mathsf{x}_1, \ldots, \mathsf{x}_k, \mathsf{y}_1, \ldots, \mathsf{y}_n) \in Form_{\mathcal{L}}$ together with $b_1, b_2, \ldots, b_n \in M$ such that

$$X = \{(a_1, a_2, \ldots, a_k) \in M^k : (\mathcal{M}, a_1, a_2, \ldots, a_k, b_1, b_2, \ldots, b_n) \vDash \varphi\}.$$

By Proposition 4.3.14, we have

$$X = \{(a_1, a_2, \ldots, a_k) \in M^k : (\mathcal{M}_{exp}, a_1, a_2, \ldots, a_k, b_1, b_2, \ldots, b_n) \vDash \varphi\}.$$

Let $\alpha \colon Var \to Term_{\mathcal{L}_M}$ be the function that codes simultaneous substitution of the (closed terms) $\underline{b_i}$ for the y_i. By Corollary 4.6.15 together with the fact that $\underline{b_i}^{\mathcal{M}_{exp}} = b_i$ for all i, it follows that

$$X = \{(a_1, a_2, \ldots, a_k) \in M^k : (\mathcal{M}_{exp}, a_1, a_2, \ldots, a_k) \vDash \varphi^{\alpha}\}.$$

Therefore, X is definable in $\mathcal{M}_{exp}$.

Assume now that X is definable in $\mathcal{M}_{exp}$. Fix $\varphi(\mathsf{x}_1, \ldots, \mathsf{x}_k) \in Form_{\mathcal{L}_M}$ such that

$$X = \{(a_1, a_2, \ldots, a_k) \in M^k : (\mathcal{M}_{exp}, a_1, a_2, \ldots, a_k) \vDash \varphi\}.$$

Suppose that the new constants (i.e. the elements of $\mathcal{L}_M \backslash \mathcal{L}$) that appear in φ are $\underline{b_1}, \ldots, \underline{b_n}$. Let $\mathsf{y}_1, \ldots, \mathsf{y}_n$ be new variables that do not occur in φ, and let ψ be the result of simultaneously substituting the variables y_i for the corresponding $\underline{b_i}$. Using Proposition 7.1.8 applied to the languages $\mathcal{L} \subseteq \mathcal{L}_M$ and the $\mathcal{L}_M$-structure $\mathcal{M}_{exp}$, we have $\psi(\mathsf{x}_1, \ldots, \mathsf{x}_k, \mathsf{y}_1, \ldots, \mathsf{y}_n) \in Form_{\mathcal{L}}$ and also

$$(\mathcal{M}_{exp}, a_1, \ldots, a_k) \vDash \varphi \text{ if and only if } (\mathcal{M}_{exp}, a_1, \ldots, a_k, b_1, \ldots, b_n) \vDash \psi$$

whenever $a_1, \ldots, a_k \in M$. Therefore,

$$X = \{(a_1, a_2, \ldots, a_k) \in M^k : (\mathcal{M}_{exp}, a_1, \ldots, a_k, b_1, \ldots, b_n) \vDash \psi\}.$$

Since $\psi \in Form_{\mathcal{L}}$, we can use Proposition 4.3.14 to conclude that

$$X = \{(a_1, a_2, \ldots, a_k) \in M^k : (\mathcal{M}, a_1, \ldots, a_k, b_1, \ldots, b_n) \vDash \psi\}.$$

Therefore, X is definable with parameters in $\mathcal{M}$. □

7.2 Nonstandard Models of Arithmetic and Analysis

As mentioned at the beginning of the chapter, we know that every theory (in countable language) that has an infinite model has both a countable model and also an uncountable model. In particular, if $\mathcal{M}$ is an infinite structure (in a countable language), then there exists models of $Th(\mathcal{M})$ that are not isomorphic to $\mathcal{M}$. In this section, we explore what these models look like for two important structures.

We begin by working in the language $\mathcal{L} = \{0, 1, +, \cdot, <\}$ of arithmetic and with the $\mathcal{L}$-structure $\mathcal{N} = (\mathbb{N}, 0, 1, +, \cdot, <)$. Let $\mathcal{M}$ be a model of $Th(\mathcal{N})$. Notice that we have changed the role of $\mathcal{M}$ and $\mathcal{N}$ from the previous section, as it will be more natural to use $\mathcal{N}$ for the structure with universe $\mathbb{N}$. Here are some important examples of sentences that are in $Th(\mathcal{N})$, and thus must be true in $\mathcal{M}$:

1. $\underline{m} + \underline{n} = \underline{m+n}$ and $\underline{m} \cdot \underline{n} = \underline{m \cdot n}$ for all $m, n \in \mathbb{N}$.
2. $\forall \mathsf{x} \forall \mathsf{y}(\mathsf{x} + \mathsf{y} = \mathsf{y} + \mathsf{x})$.
3. $\forall \mathsf{x}(\neg(\mathsf{x} = 0) \to \exists \mathsf{y}(\mathsf{y} + 1 = \mathsf{x}))$.
4. $\forall \mathsf{x} \neg(\exists \mathsf{y}((\mathsf{x} < \mathsf{y}) \wedge (\mathsf{y} < \mathsf{x} + 1)))$.
5. For each $\varphi(\mathsf{x}) \in Form_{\mathcal{L}}$, the sentence
$$(\varphi^0_{\mathsf{x}} \wedge \forall \mathsf{x}(\varphi \to \varphi^{\mathsf{x}+1}_{\mathsf{x}})) \to \forall \mathsf{x} \varphi$$
expressing that induction holds on the (definable) subset of $\mathbb{N}$ defined by $\varphi(\mathsf{x})$ is in $Th(\mathcal{M})$. We often write this sentence in the following informal way:
$$(\varphi(0) \wedge \forall \mathsf{x}(\varphi(\mathsf{x}) \to \varphi(\mathsf{x} + 1))) \to \forall \mathsf{x} \varphi.$$

Since every element of the structure $\mathcal{N} = (\mathbb{N}, 0, 1, +, \cdot, <)$ is named by a closed term, we can use Proposition 7.1.2 to conclude that whenever $\mathcal{M}$ is a model of $Th(\mathcal{N})$, the function $h \colon \mathbb{N} \to M$ defined by letting $h(n) = \underline{n}^{\mathcal{M}}$ is an elementary embedding of $\mathcal{N}$ into $\mathcal{M}$. We also have the following simple fact.

Proposition 7.2.1. *Let $\mathcal{M}$ be a model of $Th(\mathcal{N})$, and let $h \colon \mathbb{N} \to M$ be defined by letting $h(n) = \underline{n}^{\mathcal{M}}$. The following are equivalent:*

1. *$\mathcal{M} \cong \mathcal{N}$.*
2. *h is surjective, i.e. $M = \{\underline{n}^{\mathcal{M}} : n \in \mathbb{N}\}$.*

Proof. If (2) holds, then since we already know that h is an embedding (by Proposition 7.1.2), it follows immediately that h is surjective. Suppose then that (1) holds and fix an isomorphism $f \colon \mathbb{N} \to M$ from $\mathcal{N}$ to $\mathcal{M}$. We show that we must have $f(n) = \underline{n}^{\mathcal{M}}$ for all $n \in \mathbb{N}$ by induction. Notice that
$$f(0) = f(0^{\mathcal{N}}) = 0^{\mathcal{M}}$$
and also
$$f(1) = f(1^{\mathcal{N}}) = 1^{\mathcal{M}}.$$
Now given any $n \in \mathbb{N}$ with $f(n) = \underline{n}^{\mathcal{M}}$, we have
$$\begin{aligned} f(n+1) &= f(n) +^{\mathcal{M}} f(1) \\ &= \underline{n}^{\mathcal{M}} +^{\mathcal{M}} 1^{\mathcal{M}} \\ &= (\underline{n+1})^{\mathcal{M}}, \end{aligned}$$
where the last line used the fact that $\underline{n} + 1 = \underline{n+1}$ is an element of $Th(\mathcal{N})$, and hence must be true in $\mathcal{M}$. Therefore, by induction, it follows that $f(n) = \underline{n}^{\mathcal{M}}$ for all $n \in \mathbb{N}$. Since f is surjective (because it is an isomorphism), we conclude that $M = \{\underline{n}^{\mathcal{M}} : n \in \mathbb{N}\}$. □

Definition 7.2.2. *A* nonstandard model of arithmetic *is a model $\mathcal{M}$ of $Th(\mathcal{N})$ such that $\mathcal{M} \ncong \mathcal{N}$.*

We have already seen that there are nonstandard models of arithmetic by cardinality considerations. Moreover, by Corollary 7.1.5, we also know that there exists a countable nonstandard model of arithmetic.

Suppose that $\mathcal{M}$ is a model of $Th(\mathcal{N})$. As mentioned above, the function $h\colon \mathbb{N} \to M$ defined by letting $h(n) = \underline{n}^{\mathcal{M}}$ is an elementary embedding of $\mathcal{N}$ into $\mathcal{M}$. It follows that we can view $\mathcal{M}$ as an elementary extension of $\mathcal{N}$ by identifying each $n \in \mathbb{N}$ with $\underline{n}^{\mathcal{M}}$. For the rest of our discussion, we will adopt this perspective.

Convention 7.2.3. *Given a model $\mathcal{M}$ of $Th(\mathcal{N})$, we will identify $\mathbb{N}$ with the image of $h\colon \mathbb{N} \to M$ given by $h(n) = \underline{n}^{\mathcal{M}}$, so we assume that $\mathcal{N} \preceq \mathcal{M}$.*

By Proposition 7.2.1, we know that $\mathcal{M}$ is a nonstandard model of arithmetic if and only if $\mathbb{N} \subsetneq M$ under this identification. Since $\mathcal{N} \preceq \mathcal{M}$, the restriction of $+^{\mathcal{M}}$, $\cdot^{\mathcal{M}}$, and $<^{\mathcal{M}}$ to $\mathbb{N}$ coincide with the usual addition, multiplication, and order. As it quickly becomes tiresome to write $+^{\mathcal{M}}$, $\cdot^{\mathcal{M}}$, and $<^{\mathcal{M}}$, we'll abuse notation by using just $+$, $\cdot$, and $<$ to also denote the interpretations in $\mathcal{M}$. Thus, these symbols now have two different meanings, depending on context: as formal symbols in our language and as their interpretations in $\mathcal{M}$ (which matches up with the usual notions on $\mathbb{N} \subseteq M$). Be careful to distinguish between when we are using the formal symbols, and when we are talking about the interpretation in $\mathcal{M}$.

Anything that we can express in the first-order language of $\mathcal{L}$ that is true of $\mathcal{N}$ is in $Th(\mathcal{N})$, and hence is true in any nonstandard model of arithmetic. For example, we have the following simple facts.

Proposition 7.2.4. *Let $\mathcal{M}$ be a nonstandard model of arithmetic.*

- *$+$ is associative on M.*
- *$+$ is commutative on M.*
- *$<$ is a linear ordering on M.*
- *For all $a \in M$ with $a \neq 0$, there exists $b \in M$ with $b + 1 = a$.*

Proof. Consider the following sentences:

- $\forall \mathsf{x} \forall \mathsf{y} \forall \mathsf{z}(\mathsf{x} + (\mathsf{y} + \mathsf{z}) = (\mathsf{x} + \mathsf{y}) + \mathsf{x})$.
- $\forall \mathsf{x} \forall \mathsf{y}(\mathsf{x} + \mathsf{y} = \mathsf{y} + \mathsf{x})$.
- $\forall \mathsf{x} \forall \mathsf{y} \forall \mathsf{z}((\mathsf{x} < \mathsf{y} \wedge \mathsf{y} < \mathsf{z}) \to \mathsf{x} < \mathsf{z})$.
- $\forall \mathsf{x} \neg(\mathsf{x} < \mathsf{x})$.
- $\forall \mathsf{x} \forall \mathsf{y}(\mathsf{x} < \mathsf{y} \to \neg(\mathsf{y} < \mathsf{x}))$.
- $\forall \mathsf{x} \forall \mathsf{y}(\mathsf{x} < \mathsf{y} \vee \mathsf{y} < \mathsf{x} \vee \mathsf{x} = \mathsf{y})$.
- $\forall \mathsf{x}(\mathsf{x} \neq 0 \to \exists \mathsf{y}(\mathsf{y} + 1 = \mathsf{x}))$.

Each of these sentences is true in $\mathcal{N}$, so is an element of $Th(\mathcal{N})$, and hence is true in $\mathcal{M}$. □

Given a nonstandard model of arithmetic $\mathcal{M}$, we know that there exist elements in $M \backslash \mathbb{N}$. We just showed that $<$ is a linear ordering on M, so we may wonder how these new elements relate to the elements of $\mathbb{N}$.

Proposition 7.2.5. *Let $\mathcal{M}$ be a nonstandard model of arithmetic. If $a \in M \backslash \mathbb{N}$ and $n \in \mathbb{N}$, then $n < a$.*

Proof. Let $a \in M\backslash\mathbb{N}$ and $n \in \mathbb{N}$ be arbitrary. The sentence

$$\forall \mathsf{x}\left(\mathsf{x} < \underline{n} \to \bigvee_{i=0}^{n-1} (\mathsf{x} = \underline{i})\right)$$

is in $Th(\mathcal{N})$, and hence true in $\mathcal{M}$. Since $a \in M\backslash\mathbb{N}$, we have $a \neq i$ for all $i \in \mathbb{N}$. As $<$ is a linear ordering on M, we conclude that $n < a$. □

Since the elements of $M\backslash\mathbb{N}$ are greater than all the standard natural numbers, we naturally use the following terminology.

Definition 7.2.6. *Given a nonstandard model of arithmetic $\mathcal{M}$, we let $M_{inf} = M\backslash\mathbb{N}$, and we call M_{inf} the set of* infinite *elements of $\mathcal{M}$.*

Given a nonstandard model of arithmetic $\mathcal{M}$, we know that 0 is the least element, which is followed by 1, then by 2, etc. So under the linear ordering $<$, we start with $0 < 1 < 2 < \cdots$. If $a \in M_{inf}$, then a is greater than all these elements. Now, since $a \neq 0$, we know from Proposition 7.2.4 that there exists $b \in M$ with $b + 1 = a$. Since

$$\forall \mathsf{x}(\mathsf{x} < \mathsf{x} + 1 \wedge \neg\exists \mathsf{y}(\mathsf{x} < \mathsf{y} \wedge \mathsf{y} < \mathsf{x} + 1))$$

is true in $\mathcal{N}$, it is also true in $\mathcal{M}$. In other words, every infinite element of M has an immediate predecessor under the ordering $<$, which must also be an infinite element (because if $b \in \mathbb{N}$, then $b + 1 \in \mathbb{N}$). Of course, a has an immediate successor $a + 1$ as well. Thus, in the infinite part of M, each element a gives rise (by repeating iterating immediate predecessors and successors) to a two-way discrete linear ordering that is order-isomorphic to $(\mathbb{Z}, <)$. If we naturally use $a - 1$ for the immediate predecessor of a, we obtain a structure like:

$$0 < 1 < 2 < \cdots \quad \cdots < a - 2 < a - 1 < a < a + 1 < a + 2 < \cdots$$

Now there might be other infinite elements of M, and in fact we will show that there *must* be other such elements. For example, it is reasonably straightforward to see that $a + a$ does not equal any of the above elements. In order to explore the infinite realm in more detail, we introduce the following relation.

Definition 7.2.7. *Let $\mathcal{M}$ be a nonstandard model of arithmetic. Define a relation $\sim$ on M by letting $a \sim b$ if there exists $n \in \mathbb{N}$ such that either $a + n = b$ or $b + n = a$.*

In other words, we let $a \sim b$ if a and b are "finitely" far apart. Notice that if there exists $n \in \mathbb{N}$ with $a + n = b$, then we must have $a \leq b$ in $\mathcal{M}$, because the sentence $\forall \mathsf{x}\forall \mathsf{y}((\mathsf{x} = \mathsf{x} + \mathsf{y}) \vee (\mathsf{x} < \mathsf{x} + \mathsf{y}))$ is true in $\mathcal{N}$.

Proposition 7.2.8. *If $\mathcal{M}$ is a nonstandard model of arithmetic, then $\sim$ is an equivalence relation on M.*

Proof. For any $a \in M$, we have $a + 0 = a$ because the sentence $\forall \mathsf{x}(\mathsf{x} + \mathsf{0} = \mathsf{x})$ is true in $\mathcal{N}$. Thus, $\sim$ is reflexive. Also, $\sim$ is clearly symmetric by definition. Now let $a, b, c \in M$ be arbitrary such that both $a \sim b$ and $b \sim c$. We show that $a \sim c$. There are four possible cases that fall into two categories.

- *Case 1:* Suppose that we can fix $m, n \in \mathbb{N}$ with $a + m = b$ and $b + n = c$. We then have

$$\begin{aligned} a + (m + n) &= (a + m) + n \\ &= b + n \\ &= c, \end{aligned}$$

  so as $m + n \in \mathbb{N}$, it follows that $a \sim c$.

- *Case 2:* If there exist $m, n \in \mathbb{N}$ with $b + m = a$ and $c + n = b$, the argument is similar to Case 1.

- *Case 3:* Suppose that we can fix $m, n \in \mathbb{N}$ with $a + m = b$ and $c + n = b$.
  - *Subcase 1:* Suppose that $m \leq n$. Let $k = n - m \in \mathbb{N}$. We then have

$$\begin{aligned}(c + k) + m &= c + (k + m)\\ &= c + n\\ &= b\\ &= a + m.\end{aligned}$$

    Now $\forall \mathsf{x}\forall \mathsf{y}\forall \mathsf{z}(\mathsf{x} + \mathsf{z} = \mathsf{y} + \mathsf{z} \to \mathsf{x} = \mathsf{y})$ is in $Th(\mathcal{N})$, so is true in $\mathcal{M}$. Therefore, we must have $c + k = a$, and hence $a \sim c$.
  - *Subcase 2:* Suppose that $m > n$. Let $k = n - m \in \mathbb{N}$. Following the argument in Subcase 1, it follows that $a + k = c$, so $a \sim c$.
- *Case 4:* If there exist $m, n \in \mathbb{N}$ with $b + m = a$ and $b + n = c$, the argument is similar to Case 3.

□

Definition 7.2.9. *Let $\mathcal{M}$ be a nonstandard model of arithmetic, and let $a, b \in M$. We write $a \ll b$ to mean that $a < b$ and $a \not\sim b$.*

We now show that relation $\ll$ is well-defined on the equivalence classes of $\sim$. The following lemma is useful.

Lemma 7.2.10. *Let $\mathcal{M}$ be a nonstandard model of arithmetic. Let $a, b, c \in M$ be such that $a \leq b \leq c$ and suppose that $a \sim c$. We then have $a \sim b$ and $b \sim c$.*

Proof. If either $a = b$ or $b = c$, this is trivial, so assume that $a < b < c$. Since $a < c$ and $a \sim c$, we can fix $n \in \mathbb{N}^+$ with $a + n = c$. Now the sentence

$$\forall \mathsf{x}\forall \mathsf{z}\forall \mathsf{w}(\mathsf{x} + \mathsf{w} = \mathsf{z} \to \forall \mathsf{y}((\mathsf{x} < \mathsf{y} \wedge \mathsf{y} < \mathsf{z}) \to \exists \mathsf{u}(\mathsf{u} < \mathsf{w} \wedge \mathsf{x} + \mathsf{u} = \mathsf{y})))$$

is in $Th(\mathcal{N})$, so is true in $\mathcal{M}$. Thus, we can fix $d \in M$ such that $d < n$ and $a + d = b$. Since $d < n$, it follows from Proposition 7.2.5 that $d \in \mathbb{N}$, hence $a \sim b$. The proof that $b \sim c$ is similar. □

Proposition 7.2.11. *Let $\mathcal{M}$ be a nonstandard model of arithmetic. Suppose that $a_0, b_0 \in M$ are such that $a_0 \ll b_0$. For any $a, b \in M$ with $a \sim a_0$ and $b \sim b_0$, we have $a \ll b$.*

Proof. We first show that $a < b$. Notice that $a_0 < b$, because otherwise we would have $b \leq a_0 < b_0$, so $a_0 \sim b_0$ by Lemma 7.2.10. Similarly, $a < b_0$, because otherwise we would have $a_0 < b_0 \leq a$, so $a_0 \sim b_0$ by Lemma 7.2.10. Now if $b \leq a$, then we would have

$$a_0 < b \leq a < b_0,$$

so $b \sim a_0$ by Lemma 7.2.10, hence $a_0 \sim b_0$, a contradiction. Since $<$ is a linear ordering on $\mathcal{M}$, we conclude that $a < b$.

We next show that $a \not\sim b$. If $a \sim b$, then using $a_0 \sim a$ and $b_0 \sim b$, together with the fact that $\sim$ is an equivalence relation, we can conclude that $a_0 \sim b_0$, a contradiction. Therefore, $a \not\sim b$. □

Since we have shown that $\ll$ is well-defined on the elements of the equivalence classes, we define the following ordering on the classes (where we use the notation $\overline{a}$ for the equivalence class of a).

Definition 7.2.12. *Let $\mathcal{M}$ be a nonstandard model of arithmetic. Given $a, b \in M$, we write $\overline{a} < \overline{b}$ to mean that $a \ll b$.*

The next proposition implies that there is no largest equivalence class under the ordering $<$.

Proposition 7.2.13. *Let $\mathcal{M}$ be a nonstandard model of arithmetic. For any $a \in M_{inf}$, we have $a \ll a + a$.*

Proof. Let $a \in M_{inf}$ be arbitrary. Now

$$\forall \mathsf{x}(\neg(\mathsf{x} = \mathsf{0}) \to \mathsf{x} < \mathsf{x} + \mathsf{x})$$

is true in $\mathcal{N}$, and hence is true in $\mathcal{M}$. Since $a \in M_{inf}$, we have $a \neq 0$, and thus $a < a + a$. Suppose, for the same of obtaining a contradiction, that $a \sim a + a$. Since $a < a + a$, we can fix $n \in \mathbb{N}$ with $a + n = a + a$. Since

$$\forall \mathsf{x} \forall \mathsf{y} \forall \mathsf{z}(\mathsf{x} + \mathsf{y} = \mathsf{x} + \mathsf{z} \to \mathsf{y} = \mathsf{z})$$

is true in $\mathcal{N}$, it is also true in $\mathcal{M}$. Therefore, we would have $n = a$, contradicting the fact that $a \in M_{inf}$. It follows that $a \not\sim a + a$. □

Although there is no largest equivalence class, we trivially have a smallest equivalence class $\overline{0} = \mathbb{N}$. Is there a next smallest equivalence class, i.e. is there a smallest infinite one? Intuitively, if we have an infinite element, we should be able to divide it in half to get a much "smaller" infinite element. Now $\mathbb{N}$ does not strictly have division, but it does have division with remainder.

Lemma 7.2.14. *Let $\mathcal{M}$ be a nonstandard model of arithmetic. For all $a \in M$, one of the following holds:*

1. *There exists $b \in M$ such that $a = 2 \cdot b$.*
2. *There exists $b \in M$ such that $a = 2 \cdot b + 1$.*

Proof. The sentence

$$\forall \mathsf{x} \exists \mathsf{y}(\mathsf{x} = \underline{2} \cdot \mathsf{y} \vee \mathsf{x} = \underline{2} \cdot \mathsf{y} + \underline{1})$$

is in $Th(\mathcal{N})$, and hence is true in $\mathcal{M}$. □

Proposition 7.2.15. *Let $\mathcal{M}$ be a nonstandard model of arithmetic. For any $a \in M_{inf}$, there exists $b \in M_{inf}$ with $b \ll a$.*

Proof. Let $a \in M_{inf}$ be arbitrary. Using Lemma 7.2.14, we have two cases:

- *Case 1:* Suppose first that there exists $b \in M$ such that $a = 2 \cdot b$, and fix such a b. We then have $a = b + b$ because $\forall \mathsf{x}(\underline{2} \cdot \mathsf{x} = \mathsf{x} + \mathsf{x})$ is in $Th(\mathcal{N})$. Notice that $b \notin \mathbb{N}$ because otherwise we would have $a \in \mathbb{N}$. Using Proposition 7.2.13, we conclude that $b \ll b + b = a$.
- *Case 2:* Suppose now that there exists $b \in M$ such that $a = 2 \cdot b + 1$. We then have $a = (b + b) + 1$ because $\forall \mathsf{x}(\underline{2} \cdot \mathsf{x} + \underline{1} = (\mathsf{x} + \mathsf{x}) + \underline{1})$ is in $Th(\mathcal{N})$. Notice that $b \notin \mathbb{N}$ because otherwise we would have $a \in \mathbb{N}$. By Proposition 7.2.13, we know that $b \ll b + b$. Now $b + b \sim b + b + 1$, so $b \ll b + b + 1 = a$ by Proposition 7.2.11.

□

Now we know that the infinite equivalence classes have no smallest and no largest elements. Can we have two equivalence classes that are direct neighbors? Or must there always exist an equivalence class between two others? Suppose that $a \ll b$. A natural guess for an element of an equivalence class between $\overline{a}$ and $\overline{b}$ is the average of a and b. Of course we can not necessarily divide $a + b$ by 2, but we can again use division with remainder.

Proposition 7.2.16. *Let $\mathcal{M}$ be a nonstandard model of arithmetic. For any $a, b \in M_{inf}$ with $a \ll b$, there exists $c \in M_{inf}$ with $a \ll c \ll b$.*

Proof. Let $a, b \in M_{\text{inf}}$ with $a \ll b$ be arbitrary. We again have two cases:

- *Case 1:* Suppose first that there exists $c \in M$ with $a + b = 2 \cdot c$, and fix such a c. We then have $a + b = c + c$. Since
$$\forall \mathsf{x} \forall \mathsf{y} \forall \mathsf{z}((\mathsf{x} < \mathsf{y} \wedge \mathsf{x} + \mathsf{y} = \mathsf{z} + \mathsf{z}) \to (\mathsf{x} < \mathsf{z} \wedge \mathsf{z} < \mathsf{y}))$$
is in $Th(\mathcal{N})$, it follows that $a < c < b$. Thus, we need only show that $a \not\sim c$ and $c \not\sim b$.

  Suppose that $a \sim c$. Since $a < c$, we can fix $n \in \mathbb{N}$ with $a + n = c$. We then have that
$$\begin{aligned} a + b &= c + c \\ &= a + a + 2n. \end{aligned}$$
Since additive cancellation is expressible as a first-order sentence that is true in $\mathcal{N}$, we conclude that $b = a + 2n$, contradicting the fact that $a \ll b$. Therefore $a \not\sim c$.

  Suppose that $c \sim b$. Since $c < b$, we can fix $n \in \mathbb{N}$ with $c + n = b$. We then have that
$$\begin{aligned} a + 2n + b &= a + b + 2n \\ &= c + c + n + n \\ &= (c + n) + (c + n) \\ &= b + b. \end{aligned}$$
Since additive cancellation is expressible as a first-order sentence that is true in $\mathcal{N}$, we conclude that $a + 2n = b$, contradicting the fact that $a \not\sim b$. Therefore, $b \not\sim c$.

- *Case 2:* Otherwise, there exists $c \in M$ with $a + b = 2 \cdot c + 1$. In this case, a similar argument shows that $a \ll c \ll b$.

□

As mentioned above, $\overline{0}$ is the smallest equivalence class in our ordering. If we omit this one equivalence class, then Proposition 7.2.13, Proposition 7.2.15, and Proposition 7.2.16 taken together say that the remaining equivalence classes form a dense linear ordering without endpoints. If our nonstandard model $\mathcal{M}$ is countable, then we know that this ordering of (infinite) equivalence classes is isomorphic to $(\mathbb{Q}, <)$ by Theorem 5.3.11.

Our last proposition shows how nonstandard models can simplify quantifiers. It says that asking whether a first-order statement holds for infinitely many $n \in \mathbb{N}$ is equivalent to asking whether it holds for at least one infinite element of a nonstandard model.

Proposition 7.2.17. *Let $\mathcal{M}$ be a nonstandard model of arithmetic, and let $\varphi(\mathsf{x}) \in Form_{\mathcal{L}}$. The following are equivalent:*

1. *There are infinitely many $n \in \mathbb{N}$ such that $(\mathcal{N}, n) \vDash \varphi$.*
2. *There exists $a \in M_{inf}$ such that $(\mathcal{M}, a) \vDash \varphi$.*

Proof. Suppose first that there are infinitely many $n \in \mathbb{N}$ such that $(\mathcal{N}, n) \vDash \varphi$. In this case, the sentence
$$\forall \mathsf{y} \exists \mathsf{x}(\mathsf{y} < \mathsf{x} \wedge \varphi)$$
is in $Th(\mathcal{N})$, so it holds in $\mathcal{M}$. Fixing any $b \in M_{inf}$, we may conclude that there exists $a \in M$ with $b < a$ such that $(\mathcal{M}, a) \vDash \varphi$. Since $b < a$ and $b \in M_{inf}$, we may conclude that $a \in M_{inf}$.

Conversely, suppose that there are only finitely many $n \in \mathbb{N}$ such that $(\mathcal{N}, n) \vDash \varphi$. Fix $N \in \mathbb{N}$ such that $n < N$ for all n with $(\mathcal{N}, n) \vDash \varphi$. We then have that the sentence

$$\forall \mathsf{x}(\varphi \to \mathsf{x} < \underline{N})$$

is in $Th(\mathcal{N})$, so it holds in $\mathcal{M}$. Since there is no $a \in M_{inf}$ with $a < N$ by Proposition 7.2.5, it follows that there is no $a \in M_{inf}$ such that $(\mathcal{M}, a) \vDash \varphi$. □

With a basic understanding of nonstandard models of arithmetic, let's think about nonstandard models of another theory. One of the more historically interesting ones is the theory of the ordered ring of real numbers. So let $\mathcal{L} = \{0, 1, +, -, \cdot, <\}$ be the language of ordered rings, and let $\mathcal{M} = (\mathbb{R}, 0, 1, +, -, \cdot, <)$. Now as mentioned in Section 7.1, it is *not* true that every element of M is named by a closed term, so Proposition 7.1.2 does not apply. In fact, since $\mathcal{L}$ is countable, we know that $Th(\mathcal{M})$ has a countable model, and hence models of $Th(\mathcal{M})$ need not contain (a copy of) $\mathcal{M}$ as an elementary substructure. In fact, it turns out the set of real algebraic numbers (i.e. the elements of $\mathbb{R}$ that are algebraic over $\mathbb{Q}$) forms a model of $Th(\mathcal{M})$ that is a countable elementary substructure of $\mathcal{M}$.

Of course, as we saw in Section 7.1, we should add names for the elements of $\mathbb{R}$ and work with $Diag_{el}(\mathcal{M})$ in order to obtain elementary extensions of $\mathcal{M}$. In other words, we should add a constant symbol $\underline{r}$ for each $r \in \mathbb{R}$ to the language of ordered rings. The idea is that we will have nonstandard models $Diag_{el}(\mathcal{M})$ which contain both "infinite" and "infinitesimal" elements. We can then transfer first-order statements back-and-forth, and do "calculus" in this expanded structure where the basic definitions (of say continuity) are simpler and more intuitive.

However, before carrying out this idea, we should think more carefully about the language. If we want to do calculus, then we want to have analogs of all of our favorite functions, such as sin, in the nonstandard models. Once we throw these in, it is hard to know where to draw the line. In fact, there is no reason to draw a line at all. Simply throw in relation symbols for every possible subset of $\mathbb{R}^k$, and throw in function symbols for every possible function $f \colon \mathbb{R}^k \to \mathbb{R}$. Thus, throughout the rest of this section, we work in the (gigantic) language

$$\mathcal{L} = \{0, 1, +, -, \cdot, <\} \cup \{\underline{r} : r \in \mathbb{R}\} \cup \{\underline{P} : P \subseteq \mathbb{R}^k\} \cup \{\underline{f} : f \colon \mathbb{R}^k \to \mathbb{R}\},$$

where the various $\underline{P}$ and $\underline{f}$ have the corresponding arities. Of course, we do not need to include the symbols for ordered rings, as there are corresponding symbols in the other sets, but we choose to keep them around for consistency.

Definition 7.2.18. *In the above language $\mathcal{L}$, let $\mathcal{R}$ be the structure with universe $\mathbb{R}$ and where we interpret all symbols in the natural way.*

Now clearly every element of $\mathbb{R}$ is named by the closed term $\underline{r}$ in our language $\mathcal{L}$, so Proposition 7.1.2 implies that there exists an elementary embedding of $\mathcal{R}$ into any model of $Th(\mathcal{R})$. Also, by Corollary 7.1.5, we know that $\mathcal{R}$ has a proper elementary extension. Before moving on, we first prove an analogue of Proposition 7.2.1. In this case, the proof is easier, because every element of $\mathbb{R}$ is named by a constant symbol.

Proposition 7.2.19. *Let $\mathcal{M}$ be a model of $Th(\mathcal{R})$. The following are equivalent:*

1. $\mathcal{M} \cong \mathcal{R}$.
2. $M = \{\underline{r}^{\mathcal{M}} : r \in \mathbb{R}\}$.

Proof. If (2) holds, then the h of Proposition 7.1.2 is surjective and hence an isomorphism. Suppose then that (1) holds and fix an isomorphism $f \colon \mathbb{R} \to \mathcal{M}$ from $\mathcal{R}$ to $\mathcal{M}$. For any $r \in \mathbb{R}$, we must have $f(r) = f(\underline{r}^{\mathcal{R}}) = \underline{r}^{\mathcal{M}}$. Since f is surjective, it follows that $M = \{\underline{r}^{\mathcal{M}} : r \in \mathbb{R}\}$. □

Definition 7.2.20. *A* nonstandard model of analysis *is a model $\mathcal{M}$ of $Th(\mathcal{R})$ such that $\mathcal{M} \ncong \mathcal{R}$.*

As mentioned above, we can view any model $\mathcal{M}$ of $Th(\mathcal{R})$ as an elementary extension of $\mathbb{R}$ by identifying $r \in \mathbb{R}$ with $\underline{r}^{\mathcal{M}}$. As in the case for nonstandard models of arithmetic, we will use $+$, $\cdot$, $-$, and $<$ in place of $+^{\mathcal{M}}$, $\cdot^{\mathcal{M}}$, $-^{\mathcal{M}}$, and $<^{\mathcal{M}}$. Since $\mathcal{R} \preceq \mathcal{M}$, the restriction of these interpretations match up with the usual notions on $\mathbb{R}$, so there is no confusion on the semantic side. Now we also have function symbols like $\underline{\sin}$ in our language. We could similarly use $\sin(a)$ in place of $\underline{\sin}^{\mathcal{M}}(a)$ for elements of $a \in M \backslash \mathbb{R}$, but it is typical in the field the adopt the following notation to avoid conflating every function and relation.

Notation 7.2.21. *For the rest of this section, fix a nonstandard model of analysis and denote it by ${}^*\mathcal{R}$, and its universe by ${}^*\mathbb{R}$. We know that $h\colon \mathbb{R} \to {}^*\mathbb{R}$ defined by $h(r) = \underline{r}^{{}^*\mathcal{R}}$ is an elementary embedding, and we identify each $r \in \mathbb{R}$ with its image so that $\mathcal{R} \preceq {}^*\mathcal{R}$. Instead of writing $\underline{f}^{{}^*\mathcal{R}}$ for each $f\colon \mathbb{R}^k \to \mathbb{R}$, we simply write ${}^*f$. We also use similar notation for each $P \subseteq \mathbb{R}^k$. Finally, for operations like $+$ and $\cdot$, we will abuse notation and omit the $*$'s.*

Thus, we think of the function $\sin\colon \mathbb{R} \to \mathbb{R}$ as extending to a function ${}^*\sin\colon {}^*\mathbb{R} \to {}^*\mathbb{R}$. Since $\mathcal{R}$ is an elementary substructure of ${}^*\mathcal{R}$, we know that ${}^*\sin(r) = \sin(r)$ for each $r \in \mathbb{R}$, so ${}^*\sin \restriction \mathbb{R}$ is just the function sin. Moreover, this is true for *every* function and relation on $\mathbb{R}$.

Since ${}^*\mathcal{R}$ is a model of $Th(\mathcal{R})$, we know that every first-order sentence that is true in $\mathbb{R}$ will also be true in the proper elementary extension ${}^*\mathcal{R}$. For example, we immediately obtain the following.

Proposition 7.2.22. *${}^*\mathcal{R}$ is an ordered field.*

Proof. The ordered field axioms can expressed as first-order sentences, each of which is true in $\mathcal{R}$. □

Before moving on, we should say a bit more about multiplicative inverse in the field ${}^*\mathcal{R}$. As alluded to in the proof, the sentence $\forall \mathsf{x}(\neg(\mathsf{x} = \mathsf{0}) \to \exists \mathsf{y}(\mathsf{x} \cdot \mathsf{y} = \mathsf{1}))$ is true in $\mathcal{R}$ and hence in ${}^*\mathcal{R}$. Moreover, we know that multiplicative inverses of nonzero elements are unique, as that is true in any field. Now given $a \in \mathbb{R}$ with $a \neq 0$, we can naturally write $\frac{1}{a}$ for the multiplicative inverse of a. However, we can also get our hands on the multiplicative inverse of a directly using one of the function symbols. Let $f\colon \mathbb{R} \to \mathbb{R}$ be the function

$$f(r) = \begin{cases} \frac{1}{r} & \text{if } r \neq 0 \\ 0 & \text{otherwise.} \end{cases}$$

Now $\underline{f} \in \mathcal{L}$, so ${}^*f$ is a function from ${}^*\mathbb{R}$ to ${}^*\mathbb{R}$. Moreover, the sentence

$$\forall \mathsf{x}(\neg(\mathsf{x} = \mathsf{0}) \to \mathsf{x} \cdot \underline{f}\mathsf{x} = \mathsf{1})$$

is true in $\mathcal{R}$, and hence is also true in ${}^*\mathcal{R}$. Thus, we have $\frac{1}{a} = {}^*f(a)$ for all $a \in {}^*\mathbb{R} \backslash \{0\}$. The ability to use f as a way to obtain the multiplicative inverse of an element is useful, as it is typically easier to write sentences using $\underline{f}$ rather than having to use quantifiers to refer to the inverse of an element.

With that background in hand, we can ask the same question that we did for nonstandard models of arithmetic. We know that there exist elements of ${}^*\mathbb{R} \backslash \mathbb{R}$, and we know that $<$ is a linear ordering on ${}^*\mathbb{R}$. Where do the new elements live in the ordering? As we will see, we can find them sprinkled throughout. We start by showing that there must exist a positive element that is less than every positive element of $\mathbb{R}$, i.e. the structure ${}^*\mathcal{R}$ contains an infinitesimal.

Proposition 7.2.23. *There exists $z \in {}^*\mathbb{R}$ such that $z > 0$ and $z < \varepsilon$ for all $\varepsilon \in \mathbb{R}$ with $\varepsilon > 0$.*

Proof. Fix $a \in {}^*\mathbb{R} \backslash \mathbb{R}$. We have the following cases:

- *Case 1:* Suppose that $a > r$ for all $r \in \mathbb{R}$. We claim that $\frac{1}{a}$ works as such a z. One way to see this is use the general fact that whenever $0 < b < c$ in an ordered field, then $0 < \frac{1}{c} < \frac{1}{b}$. Alternatively, we can work directly with sentences as follows. Let $f \colon \mathbb{R} \to \mathbb{R}$ be the above function

$$f(r) = \begin{cases} \frac{1}{r} & \text{if } r \neq 0 \\ 0 & \text{otherwise,} \end{cases}$$

  and let $z = {}^*f(a)$. We then have have that $z > 0$ using the sentence

$$\forall \mathsf{x}(\underline{0} < \mathsf{x} \to \underline{0} < \underline{f}\mathsf{x}).$$

  Also, for any $\varepsilon \in \mathbb{R}$ with $\varepsilon > 0$, we have that $a > \frac{1}{\varepsilon}$ by assumption, so $z < \varepsilon$ using the sentence

$$\forall \mathsf{x}(\underline{f}\ \underline{\varepsilon} < \mathsf{x} \to \underline{f}\mathsf{x} < \underline{\varepsilon}).$$

- *Case 2:* Suppose that $a < r$ for all $r \in \mathbb{R}$. We then have that $a < -r$ for all $r \in \mathbb{R}$ and hence $r < -a$ for all $r \in \mathbb{R}$. Thus, we may take $z = {}^*f(-a)$ by the argument in Case 1.

- *Case 3:* Suppose then that there exists $r \in \mathbb{R}$ with $r < a$ and there exists $r \in \mathbb{R}$ with $a < r$. Let

$$X = \{r \in \mathbb{R} : r < a\}.$$

  Notice that X is downward closed (if $r_1, r_2 \in \mathbb{R}$ with $r_2 \in X$ and $r_1 < r_2$, then $r_1 \in X$), nonempty, and bounded above. Let $s = \sup X \in \mathbb{R}$. Now $a = s$ is impossible, so either $s < a$ or $a < s$.

  - *Subcase 1:* Suppose that $s < a$. We claim that we may take $z = a - s$. Since $s < a$, we have $z = a - s > 0$. Let $\varepsilon \in \mathbb{R}$ be arbitrary with $\varepsilon > 0$. We then have that $s + \varepsilon > s = \sup X$, so $s + \varepsilon \notin X$ and hence $s + \varepsilon \geq a$. Now $s + \varepsilon \neq a$ because $s + \varepsilon \in \mathbb{R}$, so $s + \varepsilon > a$. It follows that $z = a - s < \varepsilon$.
  - *Subcase 2:* Suppose that $a < s$. We claim that we may take $z = s - a$. Since $a < s$, we have $z = s - a > 0$. Let $\varepsilon \in \mathbb{R}$ be arbitrary with $\varepsilon > 0$. We then that $s - \varepsilon < s = \sup X$, so we may fix $r \in X$ with $s - \varepsilon < r$. Since X is downward closed, we have that $s - \varepsilon \in X$, so $s - \varepsilon < a$. It follows that $z = s - a < \varepsilon$.

□

Now that we know that there exists an infinitesimal z, we can easily obtain other elements of ${}^*\mathbb{R}\backslash\mathbb{R}$. For example, since $z < 1$ and $z > 0$, we know that $0 < z^2 < z$, so there are other positive infinitesimals. Negative infinitesimals like $-z$ also exist. A simple argument (or an appeal to the ordered field axioms) shows that $\frac{1}{z} > r$ for all $r \in \mathbb{R}$, so there exist infinite elements. Also, $3 + z$ is not infinitesimal, but $3 < 3 + z < r$ for all $r \in \mathbb{R}$ with $r > 3$.

Of course, we have a symbol for the absolute value function $g \colon \mathbb{R} \to \mathbb{R}$. Since

$$\underline{g}\mathsf{0} = \mathsf{0} \wedge \forall \mathsf{x}(\neg(\mathsf{x} = \mathsf{0}) \to \mathsf{0} < \underline{g}\mathsf{x})$$

is true in $\mathcal{R}$, it is also true in ${}^*\mathcal{R}$. Other basic properties like ${}^*g(a) = -a$ whenever $a < 0$, and the triangle inequality ${}^*g(a + b) \leq {}^*g(a) + {}^*g(b)$, are true by writing down the corresponding sentences. We will simply use $|a|$ in place of ${}^*g(a)$.

Definition 7.2.24.

1. $\mathcal{Z} = \{a \in {}^*\mathbb{R} : |a| < \varepsilon \text{ for all } \varepsilon \in \mathbb{R} \text{ with } \varepsilon > 0\}$. *We call* $\mathcal{Z}$ *the set of* infinitesimals.

2. $\mathcal{F} = \{a \in {}^*\mathbb{R} : |a| < r \text{ for some } r \in \mathbb{R} \text{ with } r > 0\}$. *We call* $\mathcal{F}$ *the set of* finite *or* limited *elements.*

3. $\mathcal{I} = {}^*\mathbb{R} \backslash \mathcal{F}$. *We call* $\mathcal{I}$ *the set of* infinite *or* unlimited *elements.*

Proposition 7.2.25.

1. $\mathcal{Z}$ *is a subring of* ${}^*\mathbb{R}$.

2. $\mathcal{F}$ *is a subring of* ${}^*\mathbb{R}$.

3. $\mathcal{Z}$ *is a prime ideal of* $\mathcal{F}$.

Proof.

1. First notice that $0 \in \mathcal{Z}$. Let $a, b \in \mathcal{Z}$ be arbitrary. Let $\varepsilon \in \mathbb{R}$ with $\varepsilon > 0$. We have that $\frac{\varepsilon}{2} \in \mathbb{R}$ and $\frac{\varepsilon}{2} > 0$, hence $|a| < \frac{\varepsilon}{2}$ and $|b| < \frac{\varepsilon}{2}$. It follows that

$$\begin{aligned} |a + b| &\leq |a| + |b| \\ &< \frac{\varepsilon}{2} + \frac{\varepsilon}{2} \\ &= \varepsilon. \end{aligned}$$

   Therefore, $a + b \in \mathcal{Z}$. We also have that $|a| < 1$ and $|b| < \varepsilon$, hence

$$\begin{aligned} |a \cdot b| &= |a| \cdot |b| \\ &< 1 \cdot \varepsilon \\ &= \varepsilon. \end{aligned}$$

   Therefore, $a \cdot b \in \mathcal{Z}$. Finally, $\mathcal{Z}$ is clearly closed under negation.

2. Clearly, $0 \in \mathcal{F}$. Let $a, b \in \mathcal{F}$ be arbitrary. Fix $r_1, r_2 \in \mathbb{R}$ with $r_1, r_2 > 0$ such that $|a| < r_1$ and $|b| < r_2$. We have

$$\begin{aligned} |a + b| &\leq |a| + |b| \\ &< r_1 + r_2, \end{aligned}$$

   so $a + b \in \mathcal{F}$. We also have

$$\begin{aligned} |a \cdot b| &= |a| \cdot |b| \\ &< r_1 \cdot r_2 \end{aligned}$$

   so $a \cdot b \in \mathcal{F}$. Finally, $\mathcal{F}$ is clearly closed under negation.

3. We first show that $\mathcal{Z}$ is an ideal of $\mathcal{F}$. We already know from part (1) that $\mathcal{Z}$ is closed under addition and negation. Let $a \in \mathcal{F}$ and $b \in \mathcal{Z}$ be arbitrary. Fix $r \in \mathbb{R}$ with $r > 0$ and $|a| < r$. Let $\varepsilon \in \mathbb{R}$ with $\varepsilon > 0$. We then have that $\frac{\varepsilon}{r} \in \mathbb{R}$ and $\frac{\varepsilon}{r} > 0$, hence $|a| < \frac{\varepsilon}{r}$. It follows that

$$\begin{aligned} |a \cdot b| &= |a| \cdot |b| \\ &< \frac{\varepsilon}{r} \cdot r \\ &= \varepsilon. \end{aligned}$$

   Therefore, $a \cdot b \in \mathcal{Z}$. It follows that $\mathcal{Z}$ is an ideal of $\mathcal{F}$.

   We now show that $\mathcal{Z}$ is a prime ideal of $\mathcal{F}$. Let $a, b \in \mathcal{F} \backslash \mathcal{Z}$ be arbitrary. We show that $a \cdot b \notin \mathcal{Z}$. Fix $\varepsilon, \delta \in \mathbb{R}$ with $\varepsilon, \delta > 0$ such that $|a| > \varepsilon$ and $|b| > \delta$. We then have $|a \cdot b| = |a| \cdot |b| > \varepsilon \cdot \delta$, hence $a \cdot b \notin \mathcal{Z}$.

□

In contrast to the situation for nonstandard models of arithmetic, we can define two natural relations that have to do with "closeness". Given $a, b \in {}^*\mathbb{R}$, we can say that a and b are "close" if they are off by an infinitesimal, or we can use it to mean that they are off by a limited element.

Definition 7.2.26. *Let $a, b \in {}^*\mathbb{R}$.*

1. *We write $a \approx b$ to mean that $a - b \in \mathcal{Z}$.*
2. *We write $a \sim b$ to mean that $a - b \in \mathcal{F}$.*

Proposition 7.2.27. *$\approx$ and $\sim$ are equivalence relations on ${}^*\mathbb{R}$.*

Proof. Exercise using Proposition 7.2.25. □

Definition 7.2.28. *Let $a \in {}^*\mathbb{R}$. The $\approx$-equivalence class of a is called the* halo *of a. The $\sim$-equivalence class of a is called the* galaxy *of a.*

Under the relation $\approx$, we have that $\overline{0}$ is the class of infinitesimals, and $\overline{1}$ is the class of all elements of ${}^*\mathbb{R}$ that are infinitely close to 1. In contrast, under the relation $\sim$, we have that $\overline{0} = \overline{1} = \mathcal{F}$. Now similar arguments (although in some cases easier, because we just divide by 2) show that $<$ is well-defined on the equivalence classes, that there is no largest $\sim$-equivalence class, and that the $\sim$-equivalence classes are dense. Moreover, in this setting there is no smallest $\sim$-equivalence class because ${}^*\mathcal{R}$ has additive inverses. More interestingly, we have some nice algebraic properties.

Proposition 7.2.29. *Let $a_1, b_1, a_2, b_2 \in {}^*\mathbb{R}$ with $a_1 \approx b_1$ and $a_2 \approx b_2$. We have the following:*

1. *$a_1 + a_2 \approx b_1 + b_2$.*
2. *$a_1 - a_2 \approx b_1 - b_2$.*
3. *If $a_1, b_1, a_2, b_2 \in \mathcal{F}$, then $a_1 \cdot a_2 \approx b_1 \cdot b_2$.*
4. *If $a_1, b_1 \in \mathcal{F}$ and $a_2, b_2 \in \mathcal{F} \backslash \mathcal{Z}$, then $\frac{a_1}{a_2} \approx \frac{b_1}{b_2}$.*

Proof.

1. We have $a_1 - b_1 \in \mathcal{Z}$ and $a_2 - b_2 \in \mathcal{Z}$, hence
$$(a_1 + a_2) - (b_1 + b_2) = (a_1 - b_1) + (a_2 - b_2)$$
is in $\mathcal{Z}$ by Proposition 7.2.25.
2. We have $a_1 - b_1 \in \mathcal{Z}$ and $a_2 - b_2 \in \mathcal{Z}$, hence
$$(a_1 - a_2) - (b_1 - b_2) = (a_1 - b_1) - (a_2 - b_2)$$
is in $\mathcal{Z}$ by Proposition 7.2.25.
3. We have $a_1 - b_1 \in \mathcal{Z}$ and $a_2 - b_2 \in \mathcal{Z}$. Notice that
$$\begin{aligned} a_1 \cdot a_2 - b_1 \cdot b_2 &= a_1 \cdot a_2 - a_1 \cdot b_2 + a_1 \cdot b_2 - b_1 \cdot b_2 \\ &= a_1 \cdot (a_2 - b_2) + b_2 \cdot (a_1 - b_1). \end{aligned}$$
Since $a_1 \in \mathcal{F}$ and $a_2 - b_2 \in \mathcal{Z}$, we may use Proposition 7.2.25 to conclude that $a_1 \cdot (a_2 - b_2) \in \mathcal{Z}$. Similarly, we have $b_2 \cdot (a_1 - b_1) \in \mathcal{Z}$. Applying Proposition 7.2.25 again, we conclude that $a_1 \cdot a_2 - b_1 \cdot b_2 \in \mathcal{Z}$.

4. We have $a_1 - b_1 \in \mathcal{Z}$ and $a_2 - b_2 \in \mathcal{Z}$. Now

$$\begin{aligned}\frac{a_1}{a_2} - \frac{b_1}{b_2} &= \frac{a_1 \cdot b_2 - a_2 \cdot b_1}{a_2 \cdot b_2} \\ &= \frac{1}{a_2 \cdot b_2} \cdot (a_1 \cdot b_2 - a_2 \cdot b_1),\end{aligned}$$

and we know by part (3) that $a_1 \cdot b_2 - a_2 \cdot b_1 \in \mathcal{Z}$. Since $a_2, b_2 \in \mathcal{F}\backslash\mathcal{Z}$, it follows from Proposition 7.2.25 that $a_2 \cdot b_2 \in \mathcal{F}\backslash\mathcal{Z}$. Therefore, $\frac{1}{a_2 \cdot b_2} \in \mathcal{F}$ (if $\varepsilon > 0$ is such that $|a_2 \cdot b_2| > \varepsilon$, then $|\frac{1}{a_2 \cdot b_2}| < \frac{1}{\varepsilon}$), so $\frac{a_1}{a_2} - \frac{b_1}{b_2} \in \mathcal{Z}$ by Proposition 7.2.25.

□

An important fact is that every limited elements is infinitely close to a standard real number.

Proposition 7.2.30. *For every $a \in \mathcal{F}$, there exists a unique $r \in \mathbb{R}$ such that $a \approx r$.*

Proof. Fix $a \in \mathcal{F}$. We first prove existence. Let $X = \{r \in \mathbb{R} : r < a\}$ and notice that X is downward closed, nonempty, and bounded above because $a \in \mathcal{F}$. Now let $s = \sup X$ and argue as in Case 3 of Proposition 7.2.23 that $a \approx s$.

Suppose now that $r_1, r_2 \in \mathbb{R}$ are such that $a \approx r_1$ and $a \approx r_2$. We then have that $r_1 \approx r_2$ because $\approx$ is an equivalence relation. However, this is a contradiction because $|r_1 - r_2| > \frac{|r_1 - r_2|}{2} \in \mathbb{R}$. □

Definition 7.2.31. *We define a map $st\colon \mathcal{F} \to \mathbb{R}$ by letting $st(a)$ be the unique $r \in \mathbb{R}$ such that $a \approx r$. We call $st(a)$ the* standard part*, or* shadow*, of a.*

Using Proposition 7.2.25, we immediately obtain the following fact.

Corollary 7.2.32. *The function $st\colon \mathcal{F} \to \mathbb{R}$ is a surjective ring homomorphism and $ker(st) = \mathcal{Z}$. Thus, by the First Isomorphism Theorem, the quotient ring $\mathcal{F}/\mathcal{Z}$ is isomorphic to the ring $\mathbb{R}$.*

We now turn our attention to expressing Calculus concepts in the nonstandard realm by making use of infinitesimals.

Proposition 7.2.33. *Suppose that $f\colon \mathbb{R} \to \mathbb{R}$, and that $r, \ell \in \mathbb{R}$. The following are equivalent:*

1. $\lim_{x \to r} f(x) = \ell$.
2. *For all $a \approx r$ with $a \neq r$, we have ${}^*f(a) \approx \ell$.*

Proof. Suppose first that $\lim_{x \to r} f(x) = \ell$. Let $a \in {}^*\mathbb{R}\backslash\{r\}$ be arbitrary with $a \approx r$. Let $\varepsilon \in \mathbb{R}$ with $\varepsilon > 0$. Since $\lim_{x \to r} f(x) = \ell$, we may fix $\delta \in \mathbb{R}$ with $\delta > 0$ such that $|f(x) - \ell| < \varepsilon$ whenever $0 < |x - r| < \delta$. Notice that the sentence

$$\forall \mathsf{x}((0 < |\mathsf{x} - \underline{r}| \wedge |\mathsf{x} - \underline{r}| < \underline{\delta}) \to |\underline{f}(\mathsf{x}) - \underline{\ell}| < \underline{\varepsilon})$$

is in $Th(\mathcal{R}) = Th({}^*\mathcal{R})$. Now we have $a \in {}^*\mathbb{R}$ and $0 < |a - r| < \delta$, so $|{}^*f(a) - \ell| < \varepsilon$. Since ε was an arbitrary positive element of $\mathbb{R}$, it follows that ${}^*f(a) - \ell \in \mathcal{Z}$, hence ${}^*f(a) \approx \ell$.

Suppose conversely that for all $a \approx r$ with $a \neq r$, we have ${}^*f(a) \approx \ell$. Fix some $z \in \mathcal{Z}$ with $z > 0$. Let $\varepsilon \in \mathbb{R}$ with $\varepsilon > 0$. By assumption, whenever $a \in {}^*\mathbb{R}$ and $0 < |a - r| < z$, we have that ${}^*f(a) - \ell \in \mathcal{Z}$. Thus, the sentence

$$\exists \delta(\delta > 0 \wedge \forall \mathsf{x}((0 < |\mathsf{x} - \underline{r}| \wedge |\mathsf{x} - \underline{r}| < \delta) \to |\underline{f}(\mathsf{x}) - \underline{\ell}| < \underline{\varepsilon}))$$

is in $Th({}^*\mathcal{R}) = Th(\mathcal{R})$. By fixing a witnessing δ in the structure $\mathcal{R}$, we see that the limit condition holds for ε. □

Now that we have a simple condition for limits in the nonstandard rule, and a good understanding of the arithmetic of both limited and infinitesimal elements, we can give a very simple proof of the typical algebraic limit laws by transferring everything over to ${}^*\mathcal{R}$.

Proposition 7.2.34. *Suppose that $f, g\colon \mathbb{R} \to \mathbb{R}$, and that $r, \ell, m \in \mathbb{R}$. Suppose also that $\lim\limits_{x\to r} f(x) = \ell$ and $\lim\limits_{x\to r} g(x) = m$. We then have the following:*

1. $\lim\limits_{x\to r}(f+g)(x) = \ell + m$.
2. $\lim\limits_{x\to r}(f-g)(x) = \ell + m$.
3. $\lim\limits_{x\to r}(f\cdot g)(x) = \ell \cdot m$.
4. *If $m \neq 0$, then* $\lim\limits_{x\to r}(\frac{f}{g})(x) = \frac{\ell}{m}$.

Proof. In each case, we use Proposition 7.2.33. Let $a \approx r$ be arbitrary with $a \neq r$. By assumption and Proposition 7.2.33, we then have ${}^*f(a) \approx \ell$ and ${}^*g(a) \approx m$.

1. Notice that the sentence
$$\forall \mathsf{x}((\underline{f+g})\mathsf{x} = \underline{f}\mathsf{x} + \underline{g}\mathsf{x})$$
is in $Th({}^*\mathcal{R}) = Th(\mathcal{R})$. Therefore, we have
$$\begin{aligned} {}^*(f+g)(a) &= {}^*f(a) + {}^*g(a) \\ &\approx \ell + m \qquad \text{(by Proposition 7.2.29).} \end{aligned}$$
Using Proposition 7.2.33 again, we conclude that $\lim\limits_{x\to r}(f+g)(x) = \ell + m$.

2. Completely analogous to (1).

3. As in part (1), we have ${}^*(f\cdot g)(a) = {}^*f(a)\cdot {}^*g(a)$. Therefore,
$$\begin{aligned} {}^*(f+g)(a) &= {}^*f(a)\cdot {}^*g(a) \\ &\approx \ell \cdot m \qquad \text{(by Proposition 7.2.29),} \end{aligned}$$
where we are using the fact that $\ell, m, {}^*f(a), {}^*g(a) \in \mathcal{F}$, the latter two of which follow from the fact that they are infinitely close to the former two. Using Proposition 7.2.33 again, we conclude that $\lim\limits_{x\to r}(f\cdot g)(x) = \ell \cdot m$.

4. Notice that ${}^*f(a) \in \mathcal{F}$ because ${}^*f(a) \approx \ell \in \mathbb{R}$, and ${}^*g(a) \in \mathcal{F}$ because ${}^*g(a) \approx m$. We also have ${}^*g(a) \notin \mathcal{Z}$ because $m \neq 0$. Therefore,
$$\begin{aligned} {}^*\left(\frac{f}{g}\right)(a) &= \frac{{}^*f(a)}{{}^*g(a)} \\ &\approx \frac{\ell}{m} \qquad \text{(by Proposition 7.2.29).} \end{aligned}$$
Using Proposition 7.2.33 again, we conclude that $\lim\limits_{x\to r}(\frac{f}{g})(x) = \frac{\ell}{m}$.

□

Since continuity is typically defined in terms of limits, we immediately obtain the following nonstandard equivalent of continuity, which has a very intuitive feeling.

Corollary 7.2.35. *Suppose that $f : \mathbb{R} \to \mathbb{R}$, and that $r \in \mathbb{R}$. The following are equivalent:*

1. *f is continuous at r.*
2. *For all $a \approx r$, we have ${}^*f(a) \approx f(r)$.*

Of course, one could go on to find simple equivalent conditions of other analytic notions like differentiability in the nonstandard realm. By doing so, we can often simply arguments by eliminating several quantifiers and instead relying on algebraic facts.

7.3 Theories and Universal Sentences

Given any structure $\mathcal{M}$, we know that whenever $\mathcal{A}$ is an elementary substructure of $\mathcal{M}$, we have $Th(\mathcal{A}) = Th(\mathcal{M})$. However, the same is *not* true for substructures. That is, if $\mathcal{A}$ is a substructure of $\mathcal{M}$, then we often have $Th(\mathcal{A}) \neq Th(\mathcal{M})$. The one result that we have that relates truth in $\mathcal{A}$ to truth in $\mathcal{M}$ is Corollary 4.3.12. That is, any universal formula with parameters in A that is true in $\mathcal{M}$ will be true in $\mathcal{A}$, and any existential formula with parameters in A that is true in $\mathcal{A}$ will be true in $\mathcal{M}$.

When we work with sentences, then we do not have to think about the parameters. Suppose then that T is a theory. If $\mathcal{M}$ is a model of T and $\mathcal{A}$ is a substructure of $\mathcal{M}$, then we know that $\mathcal{A}$ satisfies every universal sentence in T. Our first task will be to prove the converse of this result. That is, if T is a theory, and $\mathcal{A}$ satisfies every universal sentence in T, then $\mathcal{A}$ sits inside (as a substructure) some model of T. Although it is difficult to think about how to build such a superstructure directly, Corollary 7.1.4 reduces the problem to showing that $T \cup Diag(\mathcal{A})$ is satisfiable. The key idea then is to use Compactness to argue that every finite subset is satisfiable, because any finite obstruction can be captured in terms of one universal sentence.

Proposition 7.3.1. *Let T be a theory in a language $\mathcal{L}$, and let $\Sigma = \{\tau \in T : \tau \text{ is universal}\}$. For every model $\mathcal{A}$ of Σ, there exists a model $\mathcal{M}$ of T such that $\mathcal{A}$ is a substructure of $\mathcal{M}$.*

Proof. We first show that $T \cup Diag(\mathcal{A})$ is satisfiable. Assume, for the sake of obtaining a contradiction, that $T \cup Diag(\mathcal{A})$ is not satisfiable. By Compactness, we can fix $\delta_i \in Diag(\mathcal{A})$ such that $T \cup \{\delta_1, \delta_2, \ldots, \delta_k\}$ is not satisfiable. We then have

$$T \vDash \neg(\delta_1 \wedge \delta_2 \wedge \cdots \wedge \delta_k).$$

Let $\underline{a_1}, \underline{a_2}, \ldots, \underline{a_n}$ be the new constants (i.e. the constants in $\mathcal{L}_A \backslash \mathcal{L}$) that occur in $\neg(\delta_1 \wedge \delta_2 \wedge \cdots \wedge \delta_k)$, and let $\mathsf{y}_1, \mathsf{y}_2, \ldots, \mathsf{y}_n$ be new variables that do not occur in this formula. For each i, let $\varphi_i(\mathsf{y}_1, \mathsf{y}_2, \ldots, \mathsf{y}_n) \in Form_{\mathcal{L}}$ be the result of substituting the y_j for the $\underline{a_j}$. By Proposition 7.1.8, for each i, we know that $\varphi_i \in Form_{\mathcal{L}}$ and that δ_i is the result of substituting the $\underline{a_j}$ in for the y_j. Since the $\underline{a_i}$ do not occur in T or in $\neg(\varphi_1 \wedge \varphi_2 \wedge \cdots \wedge \varphi_k)$, and T contains no free variables (every formula in T is a sentence), we can use Proposition 5.1.7 to conclude that

$$T \vDash \forall \mathsf{y}_1 \forall \mathsf{y}_2 \cdots \forall \mathsf{y}_k \neg(\varphi_1 \wedge \varphi_2 \wedge \cdots \wedge \varphi_k).$$

Notice that each φ_i is a literal because each δ_i is a literal. Thus, the formula on the right is a universal $\mathcal{L}$-sentence, and hence is an element of Σ. Since $\mathcal{A}$ is a model of Σ, it follows that

$$\mathcal{A} \vDash \forall \mathsf{y}_1 \forall \mathsf{y}_2 \cdots \forall \mathsf{y}_k \neg(\varphi_1 \wedge \varphi_2 \wedge \cdots \wedge \varphi_k).$$

Using Proposition 4.3.14, we conclude that

$$\mathcal{A}_{exp} \vDash \forall \mathsf{y}_1 \forall \mathsf{y}_2 \cdots \forall \mathsf{y}_k \neg(\varphi_1 \wedge \varphi_2 \wedge \cdots \wedge \varphi_k).$$

Therefore, we have

$$(\mathcal{A}_{exp}, a_1, a_2, \ldots, a_n) \vDash \neg(\varphi_1 \wedge \varphi_2 \wedge \cdots \wedge \varphi_k),$$

and hence we can use Corollary 4.6.15 to conclude that

$$\mathcal{A}_{exp} \vDash \neg(\delta_1 \wedge \delta_2 \wedge \cdots \wedge \delta_k).$$

However, this contradicts the fact that $\mathcal{A}_{exp} \vDash \delta_i$ for each i, as each $\delta_i \in Diag(\mathcal{A})$.

We have shown that $T \cup Diag(\mathcal{A})$ is satisfiable, so we can fix an $\mathcal{L}_A$-structure $\mathcal{N}$ that is a model of $T \cup Diag(\mathcal{A})$. Letting $\mathcal{M} = \mathcal{N} \upharpoonright \mathcal{L}$, we then have that $\mathcal{M}$ is a model of T, and that there is an embedding $h\colon A \to M$ of $\mathcal{A}$ to $\mathcal{M}$ by Corollary 7.1.4. Identifying the image of $\mathcal{A}$ under h with $\mathcal{A}$ itself, it follows that $\mathcal{A}$ is a substructure of a model of T. □

Given a theory T, the set $\{\tau \in T : \tau \text{ is universal}\}$ is not a theory. If we instead take the set of consequences of this set, then we do obtain a theory by Proposition 5.2.4.

Definition 7.3.2. *Given a theory T, we let $T_\forall = Cn(\{\tau \in T : \tau \text{ is universal}\})$.*

We can rephrase the previous result together the above discussion in terms of theories.

Corollary 7.3.3. *Let T be a theory in a language $\mathcal{L}$, and let $\mathcal{A}$ be an $\mathcal{L}$-structure. The following are equivalent:*

1. *$\mathcal{A}$ is a model of $T_\forall$.*
2. *There exists a model $\mathcal{M}$ of T such that $\mathcal{A}$ is a substructure of $\mathcal{M}$.*

Proof. If $\mathcal{A}$ is a model of $T_\forall$, then $\mathcal{A}$ is trivially a model of $\{\tau \in T : \tau \text{ is universal}\}$, so there exists a model $\mathcal{M}$ of T such that $\mathcal{A}$ is a substructure of $\mathcal{M}$ by Proposition 7.3.1.

Conversely, assume (2), and fix a model $\mathcal{M}$ of T such that $\mathcal{A}$ is a substructure of $\mathcal{M}$. For any $\tau \in T$ that is universal, we have that $\mathcal{M} \vDash \tau$, and hence $\mathcal{A} \vDash \tau$ by Corollary 4.3.12. Thus, $\mathcal{A}$ is a model of $\{\tau \in T : \tau \text{ is universal}\}$. Using Proposition 5.2.3, it follows that $\mathcal{A}$ is a model of $Cn(\{\tau \in T : \tau \text{ is universal}\}) = T_\forall$. □

Although we have obtained an elegant abstract characterization of the theory of all substructures of models of T, it is not immediately clear how to determine $T_\forall$ for a given theory T. For example, consider the theory of fields T in the language of ring theory. We know that $T = Cn(\Sigma)$ for the set Σ consisting of the following 10 sentences:

- $\forall x \forall y \forall z(x + (y + z) = (x + y) + z)$.
- $\forall x((x + 0 = x) \wedge (0 + x = x))$.
- $\forall x((x + (-x) = 0) \wedge ((-x) + x = 0))$.
- $\forall x \forall y(x + y = y + x)$.
- $\forall x((x \cdot 1 = x) \wedge (1 \cdot x = x))$.
- $\forall x \forall y \forall z(x \cdot (y \cdot z) = (x \cdot y) \cdot z)$.
- $\forall x \forall y(x \cdot y = y \cdot x)$.
- $\forall x \forall y \forall z(x \cdot (y + z) = (x \cdot y) + (x \cdot z))$.
- $\neg(0 = 1)$.
- $\forall x(\neg(x = 0) \to \exists y(x \cdot y = 1))$.

Note that all but the last of these sentences is universal. So a natural guess might be that $T_\forall$ is the set of consequences of the first 9 sentences, i.e. that $T_\forall$ is the theory of nonzero commutative rings. However, this is not the case. One way to see this is notice that not every nonzero commutative ring can be embedded in a field (e.g. $\mathbb{Z}/4\mathbb{Z}$ is not a subring of any field) and then apply Corollary 7.3.3. Such an example leads us to search for a new (interesting) universal consequence of the field axioms. An important example is the universal sentence

$$\forall x \forall y (x \cdot y = 0 \to (x = 0 \vee y = 0)),$$

which is in T because every field is an integral domain. Notice that this sentence is not a consequence of the first 9 universal sentences above as the example $\mathbb{Z}/4\mathbb{Z}$, which is not an integral domain, demonstrates. Since $T_\forall$ contains this sentence, in addition to the first 9 sentences above, it follows that $T_\forall$ contains the theory of integral domains. Are the equal? The key result we need is the following.

Corollary 7.3.4. *Let T a theory in a language $\mathcal{L}$. Suppose that $\Sigma \subseteq T$ is a set of universal sentences, and that every model of Σ is a substructure of (or can be embedded in) some model of T. Then $T_\forall = Cn(\Sigma)$.*

Proof. Since $T_\forall$ is a theory, it suffices to show that $Mod(\Sigma) = Mod(T_\forall)$ by Proposition 5.2.5. Since $\Sigma \subseteq T_\forall$ by assumption, we clearly have $Mod(T_\forall) \subseteq Mod(\Sigma)$. Conversely, given any $\mathcal{A} \in Mod(\Sigma)$, we can fix a model $\mathcal{M}$ of T that has $\mathcal{A}$ as a substructure by assumption, so $\mathcal{A} \in Mod(T_\forall)$ by (the easy direction of) Corollary 7.3.3. □

Now we know from algebra that every integral domain can be embedded in a field. Thus, letting T be the theory of fields in the language of rings, it follows that $T_\forall$ is the theory of integral domains.

Note the importance of the choice of language in the field example. Suppose that we work in the language $\mathcal{L} = \{0, 1, +, -, \cdot, -1\}$, where -1 is a unary function symbol. Now 0 does not have a multiplicative inverse in a field, but we can hack it by stipulating that $0^{-1} = 0$ by convention. If we do this, then we replace the last sentence in the field axioms by

$$\forall x((x = 0 \to x^{-1} = 0) \wedge (\neg(x = 0) \to x \cdot x^{-1} = 1).$$

Thus, in this language, every field axiom is a universal sentence. So if we let Σ' be the corresponding 10 universal sentences, then $T' = Cn(\Sigma')$ is the theory of the fields. Notice then that every substructure of a model of T' is a model of T', and we trivially have $(T')_\forall = T'$. We could instead have gone in a different direction, and worked with the smaller language $\mathcal{L} = \{0, 1, +, \cdot\}$ that omits the additive inverse symbol. In this case, we would have to change the third sentence asserting the existence of additive inverses to include an existential quantifier. In this setting, substructures are not necessarily subrings, because they do not have to be closed under additive inverses. Of course, the resulting $T_\forall$ would change as well.

We can see this more easily by looking at the theory of groups. In the (full) group theory language $\mathcal{L} = \{e, \cdot, -1\}$, the theory of groups is the set of consequences of three universal sentences, and every substructure of a group is a group. Suppose instead that we work in the restricted group theory language $\mathcal{L} = \{e, \cdot\}$, and let T be the theory of groups in this language. In this case, we have $T = Cn(\Sigma)$, where Σ is a set of three sentences, two of which are universal (associativity and properties of e), but the last one asserting the existence of additive inverses is not. In this setting, substructures of models of T are monoids, but $T_\forall$ is not the theory of monoids. For example, the universal sentences

$$\forall x \forall y \forall z((z * x = z * y) \to x = y) \quad \text{and} \quad \forall x \forall y \forall z((x * z = y * z) \to x = y)$$

asserting two-sided cancellation are both elements of $T_\forall$. However, $T_\forall$ is *not* the set of consequences of the two universal group axioms and the two universal cancellation axioms, because there does exist a cancellative monoids (i.e. a monoid that has two-sided cancellation) that can not be embedded in a group. Determining $T_\forall$ in this case is quite interesting. In contrast, every *commutative* cancellative monoid can be embedded in an abelian group, so if T is the theory of abelian groups in the restricted group theory language,then $T_\forall$ is the theory of commutative cancellative monoids.

For a final example, let $\mathcal{L} = \{\mathsf{R}\}$ where R is a binary relation symbol. In Definition 5.3.10, we defined the theory DLO of dense linear orderings as the set of consequences of 6 sentences. Now the first three of these sentences are universal, and they express the axioms of a linear ordering, so $DLO_\forall$ contains the theory of linear orderings (and, of course, every substructure of a dense linear ordering is a linear ordering). In contrast, the last three sentences are not universal. In this case, it turns out that $DLO_\forall$ is the theory of linear orderings, because every linear ordering can be embedding in a dense linear ordering, essentially by putting a copy of $\mathbb{Q}$ at both ends, and a copy of $\mathbb{Q}$ between any two immediate neighbors.

7.4 Model Theoretic Characterization of Quantifier Elimination

We now prove the converse to Proposition 5.5.8.

Theorem 7.4.1. *Let T be a theory. Suppose that $n \geq 1$ and that $\varphi(\mathsf{x}_1, \ldots, \mathsf{x}_n) \in Form_\mathcal{L}$. The following are equivalent:*

1. *There exists a quantifier-free $\psi(\mathsf{x}_1, \ldots, \mathsf{x}_n) \in Form_\mathcal{L}$ such that $T \vDash \varphi \leftrightarrow \psi$.*
2. *Whenever $\mathcal{M}$ and $\mathcal{N}$ are models of T sharing a common substructure $\mathcal{A}$ (which need* not *be a model of T), and $a_1, a_2, \ldots, a_n \in A$, we have that $(\mathcal{M}, a_1, \ldots, a_n) \vDash \varphi$ if and only if $(\mathcal{N}, a_1, \ldots, a_n) \vDash \varphi$.*

Proof. Proposition 5.5.8 gives the direction (1) $\Rightarrow$ (2). We prove (2) $\Rightarrow$ (1). We may assume, by changing the names of bound variables if necessary, that $\mathsf{x}_i \notin BoundVar(\varphi)$ for each i. Let

$$\Gamma = \{\gamma(\mathsf{x}_1, \ldots, \mathsf{x}_n) \in Form_\mathcal{L} : \gamma \text{ is quantifier-free and } T \vDash \varphi \rightarrow \gamma\}.$$

The essential part of the argument is showing that $T \cup \Gamma \vDash \varphi$, which we do below. We first show how to finish the argument under this assumption. Suppose then that $T \cup \Gamma \vDash \varphi$. By Compactness, we can fix a finite set $\{\gamma_1, \ldots, \gamma_m\} \subseteq \Gamma$ such that $T \cup \{\gamma_1, \ldots, \gamma_m\} \vDash \varphi$. We then have

$$T \vDash (\gamma_1 \wedge \cdots \wedge \gamma_m) \rightarrow \varphi.$$

Since $T \vDash \varphi \rightarrow \gamma_i$ for each i, it follows that

$$T \vDash \varphi \leftrightarrow (\gamma_1 \wedge \cdots \wedge \gamma_m),$$

so we can let $\psi = \gamma_1 \wedge \cdots \wedge \gamma_m$.

We now show that $T \cup \Gamma \vDash \varphi$. Let $(\mathcal{M}, a_1, a_2, \ldots, a_n)$ be an arbitrary model of $T \cup \Gamma$. To show that $(\mathcal{M}, a_1, a_2, \ldots, a_n) \vDash \varphi$, we instead show that there exists a model $\mathcal{N}$ of T containing the substructure generated by $\{a_1, a_2, \ldots, a_n\}$ such that $(\mathcal{N}, a_1, a_2, \ldots, a_n) \vDash \varphi$, and use (2).

Let $\mathcal{A}$ be the substructure of $\mathcal{M}$ generated by $\{a_1, a_2, \ldots, a_n\}$, let $\underline{a_1}, \underline{a_2}, \ldots, \underline{a_n}$ be new constant symbols, and let $\mathcal{L}' = \mathcal{L} \cup \{\underline{a_1}, \underline{a_2}, \ldots, \underline{a_n}\}$. Let $\mathcal{A}'$ be the $\mathcal{L}'$-structure that is obtained by expanding the $\mathcal{L}$-structure $\mathcal{A}$ so that $\underline{a_j}^{\mathcal{A}'} = a_j$ for all j. Since $\mathcal{A}$ is generated by $\{a_1, a_2, \ldots, a_n\}$, it follows that every element of $\mathcal{A}'$ is named by a closed $\mathcal{L}'$-term. Let

$$\Sigma = \{\sigma \in Sent_{\mathcal{L}'} : \sigma \in Literal_{\mathcal{L}'} \text{ and } \mathcal{A}' \vDash \sigma\},$$

and let $\tau \in Sent'_\mathcal{L}$ be the result of plugging in the $\underline{a_j}$ for the x_j.

We claim that $T \cup \Sigma \cup \{\tau\}$ is satisfiable. Assume, for the sake of obtaining a contradiction, that it is not satisfiable. By Compactness, we can fix $\sigma_1, \sigma_2, \ldots, \sigma_k \in \Sigma$ such that

$$T \cup \{\sigma_1, \sigma_2, \ldots, \sigma_k\} \cup \{\tau\}$$

is not satisfiable. We then have

$$T \vDash \tau \to \neg(\sigma_1 \wedge \sigma_2 \wedge \cdots \wedge \sigma_k).$$

Notice that $\mathsf{x}_1, \mathsf{x}_2, \ldots, \mathsf{x}_n$ do not occur in $\tau \to \neg(\sigma_1 \wedge \sigma_2 \wedge \cdots \wedge \sigma_k)$. For each i, let $\psi_i(\mathsf{x}_1, \mathsf{x}_2, \ldots, \mathsf{x}_n) \in Form_{\mathcal{L}}$ be the result of substituting the x_j for the a_j in the sentence σ_i. By Proposition 7.1.8, for each i, we know that $\psi_i \in Form_{\mathcal{L}}$ and that σ_i is the result of substituting the a_j in for the x_j. Since the a_j do not occur in T or in $\varphi \to \neg(\psi_1 \wedge \psi_2 \wedge \cdots \wedge \psi_k)$, and T contains no free variables (as every formula in T is a sentence), we can use Proposition 5.1.7 to conclude that

$$T \vDash \varphi \to \neg(\psi_1 \wedge \psi_2 \wedge \cdots \wedge \psi_k).$$

Thus, $\neg(\psi_1 \wedge \psi_2 \wedge \cdots \wedge \psi_k) \in \Gamma$. Recall that $(\mathcal{M}, a_1, a_2, \ldots, a_n)$ is a model of Γ, so

$$(\mathcal{M}, a_1, a_2, \ldots, a_n) \vDash \neg(\psi_1 \wedge \psi_2 \wedge \cdots \wedge \psi_k).$$

Since $\neg(\psi_1 \wedge \psi_2 \wedge \cdots \wedge \psi_k)$ is quantifier-free and $\mathcal{A}$ is a substructure of $\mathcal{M}$, it follows from Corollary 4.3.12 that

$$(\mathcal{A}, a_1, a_2, \ldots, a_n) \vDash \neg(\psi_1 \wedge \psi_2 \wedge \cdots \wedge \psi_k).$$

However, this is a contradiction because $\mathcal{A}' \vDash \sigma_i$ for each i, and so $(\mathcal{A}, a_1, a_2, \ldots, a_n) \vDash \psi_i$ for each i by Proposition 7.1.8.

We have shown that $T \cup \Sigma \cup \{\tau\}$ is satisfiable, so we can fix an $\mathcal{L}'$-structure $\mathcal{N}'$ that is a model of $T \cup \Sigma \cup \{\tau\}$. Since every element of $\mathcal{A}'$ is named by a closed $\mathcal{L}'$-term, we can use Proposition 7.1.2 to conclude that there is an embedding from $\mathcal{A}'$ to $\mathcal{N}'$, and hence we can view $\mathcal{A}'$ as a substructure of $\mathcal{N}'$. Letting $\mathcal{N} = \mathcal{N}' \restriction \mathcal{L}$, we then have that $\mathcal{A}$ is a substructure of $\mathcal{N}$. Now $\mathcal{N}' \vDash \tau$, so $(\mathcal{N}, a_1, \ldots, a_n) \vDash \varphi$ by Proposition 7.1.8. By our assumption (2), we conclude that $(\mathcal{M}, a_1, \ldots, a_n) \vDash \varphi$. □

Can we use this to show that ACF has QE? Suppose that K_1 and K_2 are two algebraically closed fields that contain a common (finitely generated) substructure, which will just be a common subring R, which is necessarily an integral domain. Let F be the fraction field, and note that we can embed F in each of K_1 and K_2. Consider a formula φ that is an existential quantifier over a conjunction of literals, with finitely many parameters taken from F. Each literal is a polynomial equality or inequality with coefficients from F. Thus, φ is of the form

$$\text{There exists } x \text{ such that } (f_1(x) = 0 \wedge \cdots \wedge f_n(x) = 0 \wedge g_1(x) \neq 0 \wedge \ldots g_m(x) \neq 0).$$

We need to show that this is true in K_1 if and only if it is true in K_2. Is it true that either K_1 embeds in K_2 (in a way that is the identity on F) or vice-versa?

Key Fact: If $\mathcal{A}$ and $\mathcal{M}$ are both models of ACF, and $\mathcal{A}$ is a substructure of $\mathcal{M}$, then existential (at least over conjunction of literals) statements go down from $\mathcal{M}$ to $\mathcal{A}$.

Proposition 7.4.2. *Let $F \subseteq K$ be two algebraically closed fields, and let $f_1, \ldots, f_n, g_1, \ldots, g_m \in F[x]$. Suppose that there exists $c \in K$ such that $f_i(c) = 0$ for all i and $g_j(c) \neq 0$ for all j. Then there exists $d \in F$ such that $f_i(c) = 0$ for all i and $g_j(c) \neq 0$ for all j.*

Proof. Suppose first that some f_i is not the zero polynomial. Then c is algebraic over F, and hence must be an element of F. Otherwise, we only have g_i that are nonzero. Since there exists $c \in K$ that works, it must be the case that each g_j is nonzero. Now each g_j has only finitely many roots, and F must be infinite, so such a d exists. □

What else do we need? If F_1 and F_2 are both fields with subrings R_1 and R_2, and we have an isomorphism from R_1 to R_2, then the isomorphism can be extended to subfields.

If K_1 and K_2 are two algebraically closed fields with subfields F_1 and F_2, and we have an isomorphism from F_1 to F_2, then the isomorphism can be extended to algebraically closed subfields.

Proposition 7.4.3. *Let $\mathcal{A}$ be a substructure of $\mathcal{M}$, and assume that both $\mathcal{A}$ and $\mathcal{M}$ are models of ACF. Let $\theta(\mathsf{x}, \mathsf{y}_1, \ldots, \mathsf{y}_k) \in Form_{\mathcal{L}}$ be a conjunction of literals, and let $a_1, a_2, \ldots, a_k \in A$. If $(\mathcal{M}, a_1, \ldots, a_k) \vDash \exists \mathsf{x} \theta$ then $(\mathcal{A}, a_1, \ldots, a_k) \vDash \theta$.*

Proof. □

Why? If some f_i is nonzero, then an x that works in M is algebraic over the subfield generated by the coefficients, which lie in A, so that x must be an element of A. Otherwise, we only have g_i. If there is an x that works in $\mathcal{M}$, then all g_i must be nonzero, and so there is an x in A that works because A is infinite and each nonzero polynomial has only finitely many roots.

7.5 Exercises

1. Let $\mathcal{L} = \{\mathsf{0}, \mathsf{1}, +, \cdot, <\}$ be the language of arithmetic. For each $n \in \mathbb{N}$, let $\varphi_n(\mathsf{x})$ be the formula $\exists \mathsf{y}(\underline{n} \cdot \mathsf{y} = \mathsf{x})$.

   (a) Show that for any nonstandard model of arithmetic $\mathcal{M}$, there exists $a \in M \backslash \{\mathsf{0}^{\mathcal{M}}\}$ such that $(\mathcal{M}, a) \vDash \varphi_n$ for all $n \in \mathbb{N}^+$.

   (b) Let $P \subseteq \mathbb{N}$ be the set of primes numbers and suppose that $Q \subseteq P$. Show that there exists a countable nonstandard model of arithmetic $\mathcal{M}$ and an $a \in M$ with both of the following properties:

   - $(\mathcal{M}, a) \vDash \varphi_p$ for all $p \in Q$.
   - $(\mathcal{M}, a) \vDash \neg\varphi_p$ for all $p \in P \backslash Q$.

2. (**) Let $\mathcal{M}$ be a nonstandard model of arithmetic. Show that $\{\underline{n}^{\mathcal{M}} : n \in \mathbb{N}\} \subseteq M$ is not definable in $\mathcal{M}$.

Chapter 8

Introduction to Axiomatic Set Theory

No one shall expel us from the paradise that Cantor has created. - David Hilbert

8.1 Why Set Theory?

Set theory originated in an attempt to understand and somehow classify "small" or "negligible" sets of real numbers. Cantor's early explorations in the realm of the transfinite were motivated by a desire to understand the points of convergence of trigonometric series. The basic ideas quickly became a fundamental part of analysis.

Since then, set theory has become *a* way to unify mathematical practice, and *the* way in which mathematicians deal with the infinite in all areas of mathematics. We've all seen the proof that the set of real numbers is uncountable, but what more can be said? Exactly how uncountable is the set of real numbers? Does this taming of the infinite give us any new tools to prove interesting mathematical theorems? Is there anything more that the set-theoretic perspective provides to the mathematical toolkit other than a crude notion of size and cute diagonal arguments?

We begin by listing a few basic questions from various areas of mathematics that can only be tackled with a well-defined theory of the infinite which set theory provides.

Algebra: A fundamental result in linear algebra is that every finitely generated vector space has a basis, and any two bases have the same size. We call the unique size of any basis of a vector space the dimension of that space. Moreover, given two finitely generated vectors spaces, they are isomorphic precisely when they have the same dimension. What can be said about vector spaces that aren't finitely generated? Does every vector space have a basis? Is there a meaningful way to assign a "dimension" to every vector space in such a way that two vector spaces over the same field are isomorphic if and only if they have the same "dimension"? We need a well-defined and robust notion of infinite sets and infinite cardinality to deal with these questions.

Analysis: Lebesgue's theory of measure and integration require an important distinction between countable and uncountable sets. Aside from this use, the study of the basic structure of the Borel sets or the projective sets (an extension of the Borel sets) require some sophisticated use of set theory, in a way that can be made precise.

Foundations: A remarkable side-effect of our undertaking to systematically formalize the infinite is that we can devise a formal axiomatic and finitistic system in which virtually all of mathematical practice can be embedded in an extremely faithful manner. Whether this fact is interesting or useful depends on your philosophical stance about the nature of mathematics, but it does have an important consequence. It puts us in a position to prove that certain statements do not follow from the axioms (which have now been formally defined and are thus susceptible to a mathematical analysis), and hence can not be proven by the currently

accepted axioms. For better or worse, this feature has become the hallmark of set theory. For example, we can ask questions like:

1. Do we really need the Axiom of Choice to produce a nonmeasurable set of real numbers?
2. Is there an uncountable set of real numbers which can not be in one-to-one correspondence with the set of all real numbers?

Aside from these ideas which are applicable to other areas of mathematics, set theory is a very active area of mathematics with its own rich and beautiful structure, and deserves study for this reason alone.

8.2 Motivating the Axioms

In every modern mathematical theory (say group theory, topology, the theory of Banach spaces), we start with a list of axioms, and derive results from these. In most of the fields that we axiomatize in this way, we have several models of the axioms in mind (many different groups, many different topological spaces, etc.), and we're using the axiomatization to prove abstract results which will be applicable to each of these models. In set theory, we may think that it is our goal to study one unique universe of sets, and so our original motivation in writing down axioms is simply to state precisely what we are assuming in an area that can often be very counterintuitive. Since we will build our system in first-order logic, it turns out that there are many models of set theory as well (assuming that there is at least one...), and this is the basis for proving independence results, but this isn't our initial motivation. This section will be a little informal. We'll give the formal axioms (in a formal first-order language) and derive consequences starting in the next section.

Whether the axioms that we are writing down now are "obviously true", "correct", "justified", or even worthy of study are very interesting philosophical questions, but we will not spend much time on them here. Regardless of their epistemological status, they are now nearly universally accepted as the "right" axioms to use in the development of set theory. The objects of our theory are sets, and we have one binary relation $\in$ which represents set membership. That is, we write $x \in y$ to mean that x is an element of y. We begin with an axiom which ensures that our theory is not vacuous.

Axiom of Existence: There exists a set.

We need to have an axiom which says how equality of sets is determined in terms of the membership relation. In mathematical practice using naive set theory, the most common way to show that two sets A and B are equal is to show that each is a subset of the other. We therefore define $A \subseteq B$ to mean that for all $x \in A$, we have $x \in B$, and we want to be able to conclude that $A = B$ from the facts that $A \subseteq B$ and $B \subseteq A$. That is, we want to think of a set as being completely determined by its members, thus linking $=$ and $\in$, but we need to codify this as an axiom.

Axiom of Extensionality: For any two sets A and B, if $A \subseteq B$ and $B \subseteq A$, then $A = B$.

The Axiom of Extensionality implicitly implies a few, perhaps unexpected, consequences about the nature of sets. First, if a is a set, then we should consider the two sets $\{a\}$ and $\{a, a\}$ (if we are allowed to assert their existence) to be equal because they have the same elements. Similarly, if a and b are sets, then we should consider $\{a, b\}$ and $\{b, a\}$ to be equal. Hence, whatever a set is, it should be inherently unordered and have no notion of multiplicity. Also, since the only objects we are considering are sets, we are ruling out the existence of "atoms" other than the empty set, i.e. objects a which are not the empty set but which have no elements.

We next need some rules about how we are allowed to build sets. The naive idea is that any property we write down determines a set. That is, for any property P of sets, we may form the set $\{x : P(x)\}$. For

example, if we have a group G, we may form the center of G given by $Z(G) = \{x : x \in G \text{ and } xy = yx \text{ for all } y \in G\}$. Of course, this naive approach leads to the famous contradiction known as Russell's paradox. Let $P(x)$ be the property $x \notin x$, and let $z = \{x : P(x)\} = \{x : x \notin x\}$. We then have $z \in z$ if and only if $z \notin z$, a contradiction.

This gives our first indication that it may be in our best interest to tread carefully when giving rules about how to build sets. One now standard reaction to Russell's Paradox and other similar paradoxes in naive set theory is that the set-theoretic universe is too "large" to encapsulate into one set. Thus, we shouldn't allow ourselves the luxury of forming the set $\{x : P(x)\}$ because by doing so we may package too much into one set, and the set-theoretic universe is too "large" to make this permissible. In other words, we should only christen something as a set if it is not too "large".

However, if we already have a set A and a property P, we should be allowed to from $\{x \in A : P(x)\}$ because A is a set (hence not too "large"), so we should be allowed to assert that the subcollection consisting of those sets x *in* A such that $P(x)$ holds is in fact a set. For example, if we have a group G (so G is already known to be a set), its center $Z(G)$ is a set because $Z(G) = \{x \in G : xy = yx \text{ for all } y \in G\}$. Following this idea, we put forth the following axiom.

Axiom of Separation: For any set A and any property P of sets, we may form the set consisting of precisely those $x \in A$ such that $P(x)$, i.e. we may form the set $\{x \in A : P(x)\}$.

You may object to this axiom because of the vague notion of a "property" of sets, and that would certainly be a good point. We'll make it precise when we give the formal first-order axioms in the next section. The Axiom of Separation allows us to form sets from describable subcollections of sets we already know exist, but we currently have no way to build larger sets from smaller ones. We now give axioms which allow us to build up sets in a permissible manner.

Our first axiom along these lines will allow us to conclude that for any two sets x and y, we may put them together into a set $\{x, y\}$. Since we already have the Axiom of Separation, we will state the axiom in the (apparently) weaker form that for any two sets x and y, there is a set with both x and y as elements.

Axiom of Pairing: For any two sets x and y, there is a set A such that $x \in A$ and $y \in A$.

We next want to have an axiom which allows us to take unions. However, in mathematics, we often want to take a union over a (possibly infinite) family of sets. For example, we may have a set A_n for each natural number n, and then want to consider $\bigcup_{n\in\mathbb{N}} A_n$. By being clever, we can incorporate all of these ideas of taking unions into one axiom. The idea is the following. Suppose that we have two sets A and B, say $A = \{u, v, w\}$ and $B = \{x, z\}$. We want to be able to assert the existence of the union of A and B, which is $\{u, v, w, x, z\}$. First, by by the Axiom of Pairing, we may form the set $\mathcal{F} = \{A, B\}$, which equals $\{\{u, v, w\}, \{x, z\}\}$. Now the union of A and B is the set of elements of elements of $\mathcal{F}$. In the above example, if we can form the set $\mathcal{F} = \{A_1, A_2, A_3, \dots\}$ (later axioms will justify this), then $\bigcup_{n\in\mathbb{N}} A_n$ is the set of elements of elements of $\mathcal{F}$. Again, in the presence of the Axiom of Separation, we state this axiom in the (apparently) weaker form that for any set $\mathcal{F}$, there is set containing all elements of elements of $\mathcal{F}$.

Axiom of Union: For any set $\mathcal{F}$, there is a set U such that for all sets x, if there exists $A \in \mathcal{F}$ with $x \in A$, then $x \in U$.

We next put forward two axioms which really allow the set-theoretic universe to expand. The first is the Power Set Axiom which tells us that if we have a set A, it is permissible to form the set consisting of all subsets of A.

Axiom of Power Set: For any set A, there is a set $\mathcal{F}$ such that for all sets B, if $B \subseteq A$, then $B \in \mathcal{F}$.

Starting with the empty set $\emptyset$ (which exists using the Axiom of Existence and the Axiom of Separation), we can build a very rich collection of finite sets using the above axioms. For example, we can form $\{\emptyset\}$ using the Axiom of Pairing. We can also form $\{\emptyset\}$ by applying the Axiom of Power Set to $\emptyset$. We can then go on to form $\{\emptyset, \{\emptyset\}\}$ and many other finite sets. However, our axioms provide no means to build an infinite set.

Before getting to the Axiom of Infinity, we will lay some groundwork about ordinals. If set theory is going to serve as a basis for mathematics, we certainly need to be able to embed within it the natural numbers. It seems natural to represent the number n as some set which we think of as having n elements. Which set should we choose? Let's start from the bottom-up. The natural choice to play the role of 0 is $\emptyset$ because it is the only set without any elements. Now that we have 0, and we want 1 to be a set with one element, perhaps we should let 1 be the set $\{0\} = \{\emptyset\}$. Next, a canonical choice for a set with two elements is $\{0, 1\}$, so we let $2 = \{0, 1\} = \{\emptyset, \{\emptyset\}\}$. In general, if we have defined $0, 1, 2, \ldots, n$, we can let $n + 1 = \{0, 1, \ldots, n\}$. This way of defining the natural numbers has many advantages which we'll come to appreciate. For instance, we'll have $n < m$ if and only if $n \in m$, so we may use the membership relation to define the standard ordering of the natural numbers.

However, the $\ldots$ in the above definition of $n+1$ may make you a little nervous. Fortunately, we can give another description of $n + 1$ which avoids this unpleasantness. If we've defined n, we let $n + 1 = n \cup \{n\}$, which we can justify the existence of using the Axiom of Pairing and the Axiom of Union. The elements of $n + 1$ will then be n, and the elements of n which should "inductively" be the natural numbers up to, but not including, n.

Using the above outline, we can use our axioms to justify the existence of any particular natural number n (or, more precisely, the set that we've chosen to represent our idea of the natural number n). However, we can't justify the existence of the set of natural numbers $\{0, 1, 2, 3, \ldots\}$. To enable us to do this, we make the following definition. For any set x, let $S(x) = x \cup \{x\}$. We call $S(x)$ the *successor* of x. We want an axiom which says that there is a set containing $0 = \emptyset$ which is closed under successors.

Axiom of Infinity: There exists a set A such that $\emptyset \in A$ and for all x, if $x \in A$, then $S(x) \in A$.

With the Axiom of Infinity asserting existence, it's not too difficult to use the above axioms to show that there is a smallest (with respect to $\subseteq$) set A such that $\emptyset \in A$ and for all x, if $x \in A$, then $S(x) \in A$. Intuitively, this set is the collection of all natural numbers. Following standard set-theoretic practice, we denote this set by ω (this strange choice, as opposed to the typical $\mathbb{N}$, conforms with the standard practice of using lowercase greek letters to represent infinite ordinals).

With the set of natural numbers ω in hand, there's no reason to be timid and stop counting. We started with $0, 1, 2, \ldots$, where each new number consisted of collecting the previous numbers into a set, and we've now collected all natural numbers into a set ω. Why not continue the counting process by considering $S(\omega) = \omega \cup \{\omega\} = \{0, 1, 2, \ldots, \omega\}$? We call this set $\omega + 1$ for obvious reasons. This conceptual leap of counting into the so-called transfinite gives rise to the ordinals, the "numbers" which form the backbone of set theory.

Once we have $\omega + 1$, we can then form the set $\omega + 2 = S(\omega + 1) = \{0, 1, 2, \ldots, \omega, \omega + 1\}$, and continue on to $\omega + 3, \omega + 4$, and so on. Why stop there? If we were able to collect all of the natural numbers into a set, what's preventing us from collecting these into the set $\{0, 1, 2, \ldots, \omega, \omega + 1, \omega + 2, \ldots\}$, and continuing? Well, our current axioms are preventing us, but we shouldn't let that stand in our way. If we can form ω, surely we should have an axiom allowing us to make this new collection a set. After all, if ω isn't too "large", this set shouldn't be too "large" either since it's just another sequence of ω many sets after ω.

The same difficulty arises when you want to take the union of an infinite family of sets. In fact, the previous problem is a special case of this one, but in this generality it may feel closer to home. Suppose we have sets $A_0, A_1, A_2, \ldots$, that is, we have a set A_n for every $n \in \omega$. Of course, we should be able to justify making the union $\bigcup_{n \in \omega} A_n$ into a set. If we want to apply the Axiom of Union, we should first form the

set $\mathcal{F} = \{A_0, A_1, A_2, \dots\}$ and apply the axiom to $\mathcal{F}$. However, in general, our current axioms don't justify forming this set despite its similarity to asserting the existence of ω.

To remedy these defects, we need a new axiom. In light of the above examples, we want to say something along the lines of "if we can index a family of sets with ω, then we can form this family into a set". Using this principle, we should be able to form the set $\{\omega, \omega+1, \omega+2, \dots\}$ and hence $\{0, 1, 2, \dots, \omega, \omega+1, \omega+2, \dots\}$ is a set by the Axiom of Union. Similarly, in the second example, we should be able to form the set $\{A_0, A_1, A_2, \dots\}$. In terms of our restriction of not allowing sets to be too "large", this seems justified because if we consider ω to not be too "large", then any family of sets it indexes shouldn't be too "large" either.

There is no reason to limit our focus to ω. If we have any set A, and we can index a family of sets using A, then we should be able to assert the existence of a set containing the elements of the family. We also want to make the notion of indexing more precise, and we will do it using the currently vague notion of a property of sets as used in the Axiom of Separation.

Axiom of Collection: Suppose that A is a set and $P(x, y)$ is a property of sets such that for every $x \in A$, there is a unique set y such that $P(x, y)$ holds. Then there is a set B such that for every $x \in A$, we have $y \in B$ for the unique y such that $P(x, y)$ holds.

Our next axiom is often viewed as the most controversial due to its nonconstructive nature and the sometimes counterintuitive results it allows us to prove. I will list it here as a fundamental axiom, but we will avoid using it in the basic development of set theory below until we get to a position to see it's usefulness in mathematical practice.

The Axiom of Separation and the Axiom of Collection involved the somewhat vague notion of property, but whenever we think of a property (and the way we will make the notion of property precise using a formal language) we have a precise unambiguous definition which describes the property in mind. Our next axiom, the Axiom of Choice, asserts the existence of certain sets without the need for such a nice description. Intuitively, it says that if we have a set consisting only of nonempty sets, there is a function which picks an element out each of these nonempty sets without requiring that there be a "definable" description of such a function. We haven't defined the notion of a function in set theory, and it takes a little work to do, so we will state the axiom in the following form: For every set $\mathcal{F}$ of nonempty pairwise disjoint sets, there is a set C consisting of exactly one element from each element of $\mathcal{F}$. We think of C as a set which "chooses" an element from each of the elements of $\mathcal{F}$. Slightly more precisely, we state the axiom as follows.

Axiom of Choice: Suppose that $\mathcal{F}$ is a set such every $A \in \mathcal{F}$ is nonempty, and for every $A, B \in \mathcal{F}$, if there exists a set x with $x \in A$ and $x \in B$, then $A = B$. There exists a set C such that for every $A \in \mathcal{F}$, there is a unique $x \in C$ with $x \in A$.

Our final axiom is in no way justified by mathematical practice because it never appears in arguments outside set theory. It is also somewhat unique among our axioms in that in asserts that certain types of sets do not exist. However, adopting it gives a much clearer picture of the set-theoretic universe and it will come to play an important role in the study of set theory itself. As with the Axiom of Choice, we will avoid using it in the basic development of set theory below until we are able to see its usefulness to us.

The goal is to eliminate sets which appear circular in terms of the membership relation. For example, we want to forbid sets x such that $x \in x$ (so there is no set x such that $x = \{x\}$). Similarly, we want to forbid the existence of sets x and y such that $x \in y$ and $y \in x$. In more general terms, we don't want to have a set with an infinite descending chain each a member of the next, such as having sets x_n for each $n \in \omega$ such that $\dots \in x_2 \in x_1 \in x_0$. We codify this by saying every nonempty set A has an element which is minimal with respect to the membership relation.

Axiom of Foundation: If A is a nonempty set, then there exists $x \in A$ such that there is no set z with both $z \in A$ and $z \in x$.

8.3 Formal Axiomatic Set Theory

We now give the formal version of our axioms. We work in a first-order language $\mathcal{L}$ with a single binary relation symbol $\in$. By working in this first-order language, we are able to make precise the vague notion of property discussed above by using first-order formulas instead. However, this comes at the cost of replacing the Axiom of Separation and the Axiom of Collection by infinitely many axioms (also called an axiom scheme) since we can't quantify over formulas within the theory itself. There are other more subtle consequences of formalizing the above intuitive axioms in first-order logic which we will discuss below.

Notice also that we allow parameters (denoted by $\vec{p}$) in the Axioms of Separation and Collection so that we will be able to derive statements which universally quantified over a parameter, such as "For all groups G, the set $Z(G) = \{x \in G : xy = yx \text{ for all } x \in G\}$ exists", rather than having to reprove that $Z(G)$ is a set for each group G that we know exists. Finally, notice how we can avoid using defined notions (like $\emptyset$, $\subseteq$, and $S(\mathsf{x})$ in the Axiom of Infinity) by expanding them out into our fixed language. For example, we replace $\mathsf{x} \subseteq \mathsf{y}$ by $\forall \mathsf{w}(\mathsf{w} \in \mathsf{x} \to \mathsf{w} \in \mathsf{y})$ and replace $\emptyset \in \mathsf{z}$ by $\exists \mathsf{w}(\forall \mathsf{y}(\mathsf{y} \notin \mathsf{w}) \wedge \mathsf{w} \in \mathsf{z})$ (we could also replace it $\forall \mathsf{w}(\forall \mathsf{y}(\mathsf{y} \notin \mathsf{w}) \to \mathsf{w} \in \mathsf{z})$).

In each of the following axioms, when we write a formula $\varphi(\mathsf{x}_1, \mathsf{x}_2, \ldots, \mathsf{x}_k)$, we implicitly mean that the x_i's are distinct variables and that every free variable of φ is one of the x_i. We also use $\vec{\mathsf{p}}$ to denote a finite sequence of variables $\mathsf{p}_1, \mathsf{p}_2, \ldots, \mathsf{p}_k$. Notice that we don't need the Axiom of Existence because it is true in all $\mathcal{L}$-structures (recall that all $\mathcal{L}$-structures are nonempty).

Axiom of Extensionality:

$$\forall \mathsf{x} \forall \mathsf{y}(\forall \mathsf{w}(\mathsf{w} \in \mathsf{x} \leftrightarrow \mathsf{w} \in \mathsf{y}) \to \mathsf{x} = \mathsf{y})$$

Axiom (Scheme) of Separation: For each formula $\varphi(\mathsf{x}, \mathsf{y}, \vec{\mathsf{p}})$ we have the axiom

$$\forall \vec{\mathsf{p}} \forall \mathsf{y} \exists \mathsf{z} \forall \mathsf{x}(\mathsf{x} \in \mathsf{z} \leftrightarrow (\mathsf{x} \in \mathsf{y} \wedge \varphi(\mathsf{x}, \mathsf{y}, \vec{\mathsf{p}})))$$

Axiom of Pairing:

$$\forall \mathsf{x} \forall \mathsf{y} \exists \mathsf{z}(\mathsf{x} \in \mathsf{z} \wedge \mathsf{y} \in \mathsf{z})$$

Axiom of Union:

$$\forall \mathsf{x} \exists \mathsf{u} \forall \mathsf{z}(\exists \mathsf{y}(\mathsf{z} \in \mathsf{y} \wedge \mathsf{y} \in \mathsf{x}) \to \mathsf{z} \in \mathsf{u})$$

Axiom of Power Set:

$$\forall \mathsf{x} \exists \mathsf{z} \forall \mathsf{y}(\forall \mathsf{w}(\mathsf{w} \in \mathsf{y} \to \mathsf{w} \in \mathsf{x}) \to \mathsf{y} \in \mathsf{z})$$

Axiom of Infinity:

$$\exists \mathsf{z}(\exists \mathsf{w}(\forall \mathsf{y}(\mathsf{y} \notin \mathsf{w}) \wedge \mathsf{w} \in \mathsf{z}) \wedge \forall \mathsf{x}(\mathsf{x} \in \mathsf{z} \to \exists \mathsf{y}(\forall \mathsf{w}(\mathsf{w} \in \mathsf{y} \leftrightarrow (\mathsf{w} \in \mathsf{x} \vee \mathsf{w} = \mathsf{x})) \wedge \mathsf{y} \in \mathsf{z})))$$

Axiom (Scheme) of Collection: For each formula $\varphi(\mathsf{x}, \mathsf{y}, \vec{\mathsf{p}})$ we have the axiom

$$\forall \vec{\mathsf{p}} \forall \mathsf{w}((\forall \mathsf{x}(\mathsf{x} \in \mathsf{w} \to \exists \mathsf{y} \varphi(\mathsf{x}, \mathsf{y}, \vec{\mathsf{p}})) \wedge \forall \mathsf{x}(\mathsf{x} \in \mathsf{w} \to \forall \mathsf{u} \forall \mathsf{v}((\varphi(\mathsf{x}, \mathsf{u}, \vec{\mathsf{p}}) \wedge \varphi(\mathsf{x}, \mathsf{v}, \vec{\mathsf{p}})) \to \mathsf{u} = \mathsf{v})))$$
$$\to \exists \mathsf{z} \forall \mathsf{x}(\mathsf{x} \in \mathsf{w} \to \exists \mathsf{y}(\mathsf{y} \in \mathsf{z} \wedge \varphi(\mathsf{x}, \mathsf{y}, \vec{\mathsf{p}}))))$$

Axiom of Choice:

$$\forall \mathsf{z}((\forall \mathsf{x}(\mathsf{x} \in \mathsf{z} \to \exists \mathsf{w}(\mathsf{w} \in \mathsf{x})) \wedge \forall \mathsf{x} \forall \mathsf{y}((\mathsf{x} \in \mathsf{z} \wedge \mathsf{y} \in \mathsf{z} \wedge \exists \mathsf{w}(\mathsf{w} \in \mathsf{x} \wedge \mathsf{w} \in \mathsf{y})) \to \mathsf{x} = \mathsf{y}))$$
$$\to \exists \mathsf{c} \forall \mathsf{x}(\mathsf{x} \in \mathsf{z} \to (\exists \mathsf{w}(\mathsf{w} \in \mathsf{x} \wedge \mathsf{w} \in \mathsf{c}) \wedge \forall \mathsf{u} \forall \mathsf{v}((\mathsf{u} \in \mathsf{x} \wedge \mathsf{v} \in \mathsf{x} \wedge \mathsf{u} \in \mathsf{c} \wedge \mathsf{v} \in \mathsf{c}) \to \mathsf{u} = \mathsf{v}))))$$

Axiom of Foundation:

$$\forall \mathsf{z}(\exists \mathsf{x}(\mathsf{x} \in \mathsf{z}) \to \exists \mathsf{x}(\mathsf{x} \in \mathsf{z} \wedge \neg(\exists \mathsf{y}(\mathsf{y} \in \mathsf{z} \wedge \mathsf{y} \in \mathsf{x}))))$$

Let Ax_{ZFC} be the above set of sentences, and let ZFC $= Cn(Ax_{ZFC})$ (ZFC stands for Zermelo-Fraenkel set theory with Choice). Other presentations state the axioms of ZFC a little differently, but they all give the same theory. Some people refer to the Axiom of Separation as the Axiom of Comprehension, but Comprehension is sometimes also used to mean the contradictory statement (via Russell's Paradox) that we can always form the set $\{x : P(x)\}$, so I prefer to call it Separation. Also, some presentations refer to the Axiom of Collection as the Axiom of Replacement, but this name is more applicable to the statement that replaces the last $\to$ in the statement of Collection with a $\leftrightarrow$, and this formulation implies the Axiom of Separation.

8.4 Working from the Axioms

We have set up ZFC as a first-order theory similar to the group axioms, ring axioms, or partial orderings axioms. The fact that we have created formal first-order axioms for set theory has several far-reaching and surprising consequences. For example, if ZFC is satisfiable, then since $\mathcal{L}$ is countable, there must be a countable model of ZFC. This shocking result may seem to contradict the fact that (as we will see) ZFC proves the existence of uncountable sets. How can these statements not contradict each other? This seeming paradox, known as Skolem's paradox, can only be understood and resolved once we have a better sense of what models of ZFC look like. Notice also that if ZFC is satisfiable, then there is an uncountable model of ZFC by Proposition 6.3.4. Therefore, if ZFC has a model, then it has several nonisomorphic models. It is very natural to find all of these facts disorienting, because our original motivation was to write down axioms for "the" universe of sets.

What does a model of ZFC look like? Recall that we are working in a language $\mathcal{L}$ with just one binary relation symbol. An $\mathcal{L}$-structure $\mathcal{M}$ in this language can be visualized as a directed graph, where we draw an arrow from vertex u to vertex v if (u, v) is an element of $\in^{\mathcal{M}}$. Given a vertex v in such a directed graph, the set of *predecessors* of v is just the set of vertices that have an arrow pointing to v. From this perspective, the Axiom of Extensionality says that if two vertices have the same set of predecessors, they then must be the same vertex. The Axiom of Pairing says that given any two vertices u and w, we can always find a vertex v such that u and w are both predecessors of v. For a more interesting example, the Axiom of Separation says that given any vertex v, if we consider any subset of the predecessors of v that is definable (with parameters), then there is a vertex u whose predecessors consist of exactly this definable subset. Moreover, a defined notion like "u is a subset of w" just means that the set of predecessors of u is a subset of the set of predecessors of w.

For a concrete example, consider the $\mathcal{L}$-structure $\mathfrak{N} = (\mathbb{N}, <)$. As a directed graph, we have a vertex m for each natural number, and we have an arrow from m to n if and only if $m < n$. We determine which of the ZFC axioms are true in $\mathfrak{N}$:

- *Axiom of Extensionality:* In the structure $\mathfrak{N}$, this interprets as saying that whenever two elements of $\mathbb{N}$ have the same elements of $\mathbb{N}$ less than them, then they are equal. This holds in $\mathfrak{N}$.

- *Axiom (Scheme) of Separation:* This does not hold in $\mathfrak{N}$. Let $\varphi(\mathsf{x}, \mathsf{y})$ be the formula $\exists \mathsf{w}(\mathsf{w} \in \mathsf{x})$. The corresponding instance of Separation is:

  $$\forall \mathsf{y} \exists \mathsf{z} \forall \mathsf{x}(\mathsf{x} \in \mathsf{z} \leftrightarrow (\mathsf{x} \in \mathsf{y} \wedge \exists \mathsf{w}(\mathsf{w} \in \mathsf{x})))$$

  In the structure $\mathfrak{N}$, this interprets as saying that for all $n \in \mathbb{N}$, there is an $m \in \mathbb{N}$ such that for all $k \in \mathbb{N}$, we have $k < m$ if and only if $k < n$ and $k \neq 0$. This does not hold in $\mathfrak{N}$ because if we consider $n = 2$, there is no $m \in \mathbb{N}$ such that $0 \not< m$ and yet $1 < m$.

- *Axiom of Pairing:* In the structure $\mathfrak{N}$, this interprets as saying that whenever $m, n \in \mathbb{N}$, there exists $k \in \mathbb{N}$ such that $m < k$ and $n < k$. This holds in $\mathfrak{N}$ because given $m, n \in \mathbb{N}$, we may take $k = \max\{m, n\} + 1$.

- *Axiom of Union:* In the structure $\mathfrak{N}$, this interprets as saying that whenever $n \in \mathbb{N}$, there exists $\ell \in \mathbb{N}$ such that whenever $k \in \mathbb{N}$ has the property that there exists $m \in \mathbb{N}$ with $k < m$ and $m < n$, then $k < \ell$. This holds in $\mathfrak{N}$ because given $n \in \mathbb{N}$, we may take $\ell = n$ since if $k < m$ and $m < n$, then $k < n$ by transitivity of $<$ in $\mathbb{N}$ (in fact, we may take $\ell = n - 1$ if $n \neq 0$).

- *Axiom of Power Set:* In the structure $\mathfrak{N}$, this interprets as saying that whenever $n \in \mathbb{N}$, there exists $\ell \in \mathbb{N}$ such that whenever $m \in \mathbb{N}$ has the property that every $k < m$ also satisfies $k < n$, then $m < \ell$. This holds in $\mathfrak{N}$ because given $n \in \mathbb{N}$, we may take $\ell = n + 1$ since if $m \in \mathbb{N}$ has the property that every $k < m$ also satisfies $k < n$, then $m \leq n$ and hence $m < n + 1$.

- *Axiom of Infinity:* In the structure $\mathfrak{N}$, this interprets as saying that there exists $n \in \mathbb{N}$ such that $0 < n$ and whenever $m < n$, we have $m + 1 < n$. This does not hold in $\mathfrak{N}$.

- *Axiom (Scheme) of Collection:* This holds in $\mathfrak{N}$, as we now check. Fix a formula $\varphi(\mathsf{x}, \mathsf{y}, \vec{\mathsf{p}})$. Interpreting in $\mathfrak{N}$, we need to check that if we fix natural numbers $\vec{q}$ and an $n \in \mathbb{N}$ such that for all $k < n$ there exists a unique $\ell \in \mathbb{N}$ such that $(\mathfrak{N}, k, \ell, \vec{q}) \vDash \varphi$, then there exists $m \in \mathbb{N}$ such that for all $k < n$ there exists an $\ell < m$ such that $(\mathfrak{N}, k, \ell, \vec{q}) \vDash \varphi$. Let's then fix natural numbers $\vec{q}$ and an $n \in \mathbb{N}$, and suppose that for all $k < n$ there exists a unique $\ell \in \mathbb{N}$ such that $(\mathfrak{N}, k, \ell, \vec{q}) \vDash \varphi$. For each $k < n$, let ℓ_k be the unique element of $\mathbb{N}$ such that $(\mathfrak{N}, k, \ell_k, \vec{q}) \vDash \varphi$. Letting $m = \max\{\ell_k : k < n\} + 1$, we see that m suffices. Therefore, this holds in $\mathfrak{N}$.

- *Axiom of Choice:* In the structure $\mathfrak{N}$, this interprets as saying that whenever $n \in \mathbb{N}$ is such that

  - Every $m < n$ is nonzero.
  - For all $\ell, m < n$, there is no k with $k < \ell$ and $k < m$

  then there exists $m \in \mathbb{N}$ such that for all $k < n$, there is exactly one $\ell \in \mathbb{N}$ with $\ell < m$ and $\ell < n$. Notice that the only $n \in \mathbb{N}$ satisfying the hypothesis (that is, the above two conditions) is $n = 0$. Now for $n = 0$, the condition is trivial because we may take $m = 0$ as there is no $k < 0$. Therefore, this holds in $\mathfrak{N}$.

- *Axiom of Foundation:* In the structure $\mathfrak{N}$, this interprets as saying that whenever $n \in \mathbb{N}$ has the property that there is some $m < n$, there there exists $m < n$ such that there is no k with $k < m$ and $k < n$. Notice that $n \in \mathbb{N}$ has the property that there is some $m < n$ if and only if $n \neq 0$. Thus, this holds in $\mathfrak{N}$ because if $n \neq 0$, then we have that $0 < n$ and there is no k with $k < 0$ and $k < n$.

Is ZFC, or equivalently Ax_{ZFC}, satisfiable? Can we somehow construct a model of ZFC? These are interesting questions with subtle answers. For now, we'll just have to live with a set of axioms with no obvious models. How then do we show that an $\mathcal{L}$-sentence σ is in ZFC? Since we have two notions of implication (semantic and syntactic), we can show that either $Ax_{ZFC} \vDash \sigma$ or $Ax_{ZFC} \vdash \sigma$. Given our experience with syntactic deductions, of course we will choose the the former. When attempting to show that $Ax_{ZFC} \vDash \sigma$, we must take an arbitrary model of Ax_{ZFC} and show that it is a model of σ. Even though we do not have a simple natural example of such a model, we can still argue in this way, but we must be mindful of strange $\mathcal{L}$-structures and perhaps unexpected models.

Thus, when we develop set theory below, we will be arguing semantically via models. Rather that constantly saying "Fix a model $\mathcal{M}$ of Ax_{ZFC}" at the beginning of each proof, and proceeding by showing that $(\mathcal{M}, s) \vDash \varphi$ for various φ, we will keep the models in the background and assume that we are "living" inside one for each proof. When we are doing this, a "set" is simply an element of the universe M of

our model $\mathcal{M}$, and given two "sets" a and b, we write $a \in b$ to mean that (a, b) is an element of $\in^{\mathcal{M}}$. Notice that this approach is completely analogous to what we do in group theory. That is, when proving that statement is true in all groups, we take an arbitrary group, and "work inside" that group with the appropriate interpretations of the symbols.

Also, although there is no hierarchy of sets in our axioms, we will often follow the practice of using lowercase letters a, b, c, etc. to represent sets that we like to think of as having no internal structure (such as numbers, elements of a group, points of a topological space), use capital letters A, B, C, etc. to represent sets whose elements we like to think of as having no internal structure, and use script letters $\mathcal{A}, \mathcal{F}$, etc. to represent sets of such sets. Again, an arbitrary model of ZFC can be viewed as a directed graph, so *every* element of the model is a certain vertex (like any other) in this structure. In other words, our notational choices are just for our own human understanding.

8.5 ZFC as a Foundation for Mathematics

In the next few chapters, we'll show how to develop mathematics quite faithfully within the framework of ZFC. This raises the possibility of using set theory as a foundation for mathematical practice. However, this seems circular because our development of logic presupposed normal mathematical practice and "naive" set theory (after all, we have the *set* of axioms of ZFC). It seems that logic depends on set theory and set theory depends on logic, so how have we gained anything from a foundational perspective?

It is indeed possible, at least in principle, to get out of this vicious circle and have a completely finististic basis for mathematics. The escape is to buckle down and use syntactic arguments. In other words, we can show that $Ax_{ZFC} \vdash \sigma$ instead of showing that $Ax_{ZFC} \vDash \sigma$. Now there are infinitely many axioms of ZFC (because of the two axioms schemes), but any given deductions will only use finitely many of the axioms. Furthermore, although there are infinitely many axioms, we can mechanically check if a given $\mathcal{L}$-sentence really is an axiom. Informally, the set of axioms is a "computable" set. In this way, it would be possible in principle to make every proof completely formal and finitistic, where each line follows from previous lines by one of our proof rules. If we held ourselves to this style, then we could reduce mathematical practice to a game with finitely many symbols (if we insisted on avoiding reference to the natural numbers explicity, we could replace our infinite stock of variables Var with one variable symbol x, introduce a new symbol $'$, and refer to x_3 as x''', etc.) where each line could be mechanically checked according to our finitely many rules. Thus, it would even be possible to program a computer to check every proof.

In practice (for human beings at least), the idea of giving deductions for everything is outlandish. Leaving aside the fact that actually giving short deductions is often a painful endeavor, it turns out that even the most basic statements of mathematics, when translated into ZFC, are many thousands of symbols long, and elementary mathematical proofs (such as say the Fundamental Theorem of Arithmetic) are many thousands of lines long. We'll discuss how to develop the real numbers below, but any actual formulas talking about real numbers would be ridiculously long and incomprehensible to the human reader. Due to these reasons, and since the prospect of giving syntactic deductions for everything should give you nightmares, we will argue everything semantically in the style of any other axiomatic subject in mathematics. It is an interesting and worthwhile exercise, however, to imagine how everything could be done syntactically.

Chapter 9

Developing Basic Set Theory

9.1 First Steps

We first establish some basic set theoretic facts carefully from the axioms.

Definition 9.1.1. *If A and B are sets, we write $A \subseteq B$ to mean for all $c \in A$, we have $c \in B$.*

Although the symbol $\subseteq$ is not part of our language, we will often use $\subseteq$ in our formulas and arguments. This use is justified because it can always be transcribed into our language by replacing it with the corresponding formula as we did in the axioms. As mentioned previously, if we have a model $\mathcal{M}$ of ZFC viewed as a directed graph, and we also have $a, b \in M$, then $a \subseteq b$ will be true in M if every vertex that points to a also points to b. In other words, always remember that although we call elements "sets", in any given model, these "sets" can be viewed as vertices of a directed graph with certain properties. From this perspective, the next result says that in any model of ZFC, there is a unique vertex that has no incoming arrows.

Proposition 9.1.2. *There is a unique set with no elements.*

Proof. Fix a set b (which exists by the Axiom of Existence, or because $\mathcal{L}$-structures are nonempty by definition). By Separation applied to the formula $\mathsf{x} \neq \mathsf{x}$, there is a set c such that for all a, we have $a \in c$ if and only if $a \in b$ and $a \neq a$. Now for all sets a, we have $a = a$, hence $a \notin c$. Therefore, there is a set with no elements. If c_1 and c_2 are two sets with no elements, then by the Axiom of Extensionality, we may conclude that $c_1 = c_2$. □

Definition 9.1.3. *We use $\emptyset$ to denote the unique set with no elements.*

As above, we will often use $\emptyset$ in our formulas and arguments despite the fact that there is no constant in our language representing it. Again, this use can always be eliminated by replacing it with a formula, as we did in the Infinity, Choice, and Foundation axioms. We will continue to follow this practice without comment in the future when we introduce new definitions to stand for sets for which ZFC proves both existence and uniqueness. In each case, be sure to understand how these definitions could be eliminated.

We now show how to turn the idea of Russell's Paradox into a proof that there is no universal set.

Proposition 9.1.4. *There is no set u such that $a \in u$ for every set a.*

Proof. We prove this by contradiction. Suppose that u is a set with the property that $a \in u$ for every set a. By Separation applied to the formula $\neg(\mathsf{x} \in \mathsf{x})$, there is a set c such that for all sets a, we have $a \in c$ if and only if $a \in u$ and $a \notin a$. Since $a \in u$ for every set a, it follows that for each set a, we have $a \in c$ if and only if $a \notin a$. Therefore, $c \in c$ if and only if $c \notin c$, a contradiction. □

Proposition 9.1.5. *For all sets a and b, there is a unique set c such that, for all sets d, we have $d \in c$ if and only if either $d = a$ or $d = b$.*

Proof. Let a and b be arbitrary sets. By Pairing, there is a set e such that $a \in e$ and $b \in e$. By Separation applied to the formula $\mathsf{x} = \mathsf{a} \vee \mathsf{x} = \mathsf{b}$ (notice that we are using parameters a and b in this use of Separation), there is a set c such that for all d, we have $d \in c$ if and only if both $d \in e$ and either $d = a$ or $d = b$. It follows that $a \in c$, $b \in c$, and for any $d \in c$, we have either $d = a$ or $d = b$. Uniqueness again follows from Extensionality. □

Corollary 9.1.6. *For every set a, there is a unique set c such that, for all sets d, we have $d \in c$ if and only if $d = a$.*

Proof. Apply the previous proposition with $b = a$. □

Definition 9.1.7. *Given two sets a and b, we use the notation $\{a, b\}$ to denote the unique set guaranteed to exist by the Proposition 9.1.5. Given a set a, we use the notation $\{a\}$ to denote the unique set guaranteed to exist by the Corollary 9.1.6.*

Using the same style of argument, we can use Union and Separation to show that for every set $\mathcal{F}$, there is a unique set z consisting precisely of elements of elements of $\mathcal{F}$.

Proposition 9.1.8. *Let $\mathcal{F}$ be a set. There is a unique set U such that for all a, we have $a \in U$ if and only if there exists $B \in \mathcal{F}$ with $a \in B$.*

Proof. Exercise. □

Definition 9.1.9. *Let $\mathcal{F}$ be a set. We use the notation $\bigcup \mathcal{F}$ to denote the unique set guaranteed to exist by the previous proposition. If A and B are sets, we use the notation $A \cup B$ to denote $\bigcup\{A, B\}$.*

We now introduce some notation which conforms with the normal mathematical practice of writing sets.

Definition 9.1.10. *Suppose that $\varphi(\mathsf{x}, \mathsf{y}, \vec{\mathsf{p}})$ is a formula (in our language $\mathcal{L} = \{\in\}$), and that B and $\vec{q}$ are sets. By Separation and Extensionality, there is a unique set C such that for all sets a, we have $a \in C$ if and only if $a \in B$ and $\varphi(a, B, \vec{q})$. More formally, given a model $\mathcal{M}$ of ZFC and elements $B, \vec{q}$ of M, there is unique element C of M such that for all a, we have $a \in C$ if and only if $a \in M$ and $(\mathcal{M}, a, B, \vec{q}) \vDash \varphi$. We denote this unique set by $\{a \in B : \varphi(a, B, \vec{q})\}$.*

With unions in hand, what about intersections? As in unions, the general case to consider is when we have a set $\mathcal{F}$, which we think of as a family of sets. We then want to collect those a such that $a \in B$ for all $B \in \mathcal{F}$ into a set. However, we do need to be a little careful. What happens if $\mathcal{F} = \emptyset$? It seems that our definition would want to make the the intersection of the sets in $\mathcal{F}$ consists of all sets, contrary to Proposition 9.1.4. However, this is the only case which gives difficulty, because if $\mathcal{F} \neq \emptyset$, we can take the intersection to be a subset of one (any) of the elements of $\mathcal{F}$.

Proposition 9.1.11. *Let $\mathcal{F}$ be a set with $\mathcal{F} \neq \emptyset$. There is a unique set I such that for all a, we have $a \in I$ if and only if $a \in B$ for all $B \in \mathcal{F}$.*

Proof. Since $\mathcal{F} \neq \emptyset$, we may fix $C \in \mathcal{F}$. Let $I = \{a \in C : \forall B(B \in \mathcal{F} \to a \in B)\}$. For all a, we have $a \in I$ if and only if $a \in B$ for all $B \in \mathcal{F}$. Uniqueness again follows from Extensionality. □

Definition 9.1.12. *Let $\mathcal{F}$ be a set with $\mathcal{F} \neq \emptyset$. We use the notation $\bigcap \mathcal{F}$ to denote the unique set guaranteed to exist by the previous proposition. If A and B are sets, we use the notation $A \cap B$ to denote $\bigcap\{A, B\}$.*

If A is a set, then we can not expect the complement of A to be a set because the union of such a purported set with A would be a set which has every set as an element, contrary to Proposition 9.1.4. However, if A and B are sets, and $A \subseteq B$, we can take the relative complement of A in B.

Proposition 9.1.13. *Let A and B be sets with $A \subseteq B$. There is a unique set C such that for all a, we have $a \in C$ if and only if $a \in B$ and $a \notin A$.*

Proof. Exercise. □

Definition 9.1.14. *Let A and B be sets with $A \subseteq B$. We use the notation $B\backslash A$, or $B - A$, to denote the unique set guaranteed to exist by the previous proposition.*

Since sets have no internal order to them, we need a way to represent ordered pairs. Fortunately (since it means we don't have to extend our notion of set), there is a hack that allows us to build sets capturing the only essential property of an ordered pair.

Definition 9.1.15. *Given two sets a and b, we let $(a, b) = \{\{a\}, \{a, b\}\}$.*

Proposition 9.1.16. *Let a, b, c, d be sets. If $(a, b) = (c, d)$, then $a = c$ and $b = d$.*

Proof. Let a, b, c, d be arbitrary sets with $\{\{a\}, \{a, b\}\} = \{\{c\}, \{c, d\}\}$. We first show that $a = c$. Since $\{c\} \in \{\{a\}, \{a, b\}\}$, either $\{c\} = \{a\}$ or $\{c\} = \{a, b\}$. In either case, we have $a \in \{c\}$, hence $a = c$. We now need only show that $b = d$. Suppose instead that $b \neq d$. Since $\{a, b\} \in \{\{c\}, \{c, d\}\}$, we have either $\{a, b\} = \{c\}$ or $\{a, b\} = \{c, d\}$. In either case, we conclude that $b = c$ (because either $b \in \{c\}$, or $b \in \{c, d\}$ and $b \neq d$). Similarly, since $\{c, d\} \in \{\{a\}, \{a, b\}\}$, we have either $\{c, d\} = \{a\}$ or $\{c, d\} = \{a, b\}$, and in either case we conclude that $d = a$. Therefore, using the fact that $a = c$, it follows that $b = d$. □

We next turn to Cartesian products. Given two sets A and B, we would like to form the set $\{(a, b) : a \in A$ and $b \in B\}$. Justifying that we can collect these elements into a set takes a little work, since we don't have a set that we are "carving" out from. The idea is as follows. For each fixed $a \in A$, we can assert the existence of $\{a\} \times B = \{(a, b) : b \in B\}$ using Collection (and Separation) because B is a set. Then using Collection (and Separation) again, we can assert the existence of $\{\{a\} \times B : a \in A\}$ since A is a set. The Cartesian product is then the union of this set. At later points, we will consider this argument sufficient, but we give a slightly more formal version here to really see how the axioms of Collection and Separation are applied and where the formulas come into play.

Proposition 9.1.17. *For any two sets A and B, there exists a unique set, denoted by $A \times B$, such that for all x, we have $x \in A \times B$ if and only if there exists $a \in A$ and $b \in B$ with $x = (a, b)$.*

Proof. Let $\varphi(\mathsf{x}, \mathsf{a}, \mathsf{b})$ be a formula expressing that "$\mathsf{x} = (\mathsf{a}, \mathsf{b})$" (think about how to write this down). Letting $\exists!$ be shorthand for "there is a unique", the following sentence follows (either semantically or syntactically) from Ax_{ZFC}, and hence is an element of ZFC:

$$\forall \mathsf{a} \forall \mathsf{B} (\forall \mathsf{b} (\mathsf{b} \in \mathsf{B} \rightarrow \exists! \mathsf{x} \varphi(\mathsf{x}, \mathsf{a}, \mathsf{b}))).$$

Therefore, by Collection, we may conclude that the following sentence is an element of ZFC:

$$\forall \mathsf{a} \forall \mathsf{B} \exists \mathsf{C} \forall \mathsf{b} (\mathsf{b} \in \mathsf{B} \rightarrow \exists \mathsf{x} (\mathsf{x} \in \mathsf{C} \wedge \varphi(\mathsf{x}, \mathsf{a}, \mathsf{b}))).$$

Next using Separation and Extensionality, we have the following:

$$\forall \mathsf{a} \forall \mathsf{B} \exists! \mathsf{C} \forall \mathsf{b} (\mathsf{b} \in \mathsf{B} \leftrightarrow \exists \mathsf{x} (\mathsf{x} \in \mathsf{C} \wedge \varphi(\mathsf{x}, \mathsf{a}, \mathsf{b}))).$$

From here, it follows that

$$\forall \mathsf{A} \forall \mathsf{B} \forall \mathsf{a} (\mathsf{a} \in \mathsf{A} \rightarrow \exists! \mathsf{C} \forall \mathsf{b} (\mathsf{b} \in \mathsf{B} \leftrightarrow \exists \mathsf{x} (\mathsf{x} \in \mathsf{C} \wedge \varphi(\mathsf{x}, \mathsf{a}, \mathsf{b})))).$$

Using Collection again, we may conclude that

$$\forall \mathsf{A} \forall \mathsf{B} \exists \mathcal{F} \forall \mathsf{a} (\mathsf{a} \in \mathsf{A} \rightarrow \exists \mathsf{C} (\mathsf{C} \in \mathcal{F} \wedge \forall \mathsf{b} (\mathsf{b} \in \mathsf{B} \leftrightarrow \exists \mathsf{x} (\mathsf{x} \in \mathsf{C} \wedge \varphi(\mathsf{x}, \mathsf{a}, \mathsf{b}))))),$$

and hence the following is an element of ZFC:

$$\forall \mathsf{A} \forall \mathsf{B} \exists \mathcal{F} \forall \mathsf{a} \forall \mathsf{b}((\mathsf{a} \in \mathsf{A} \wedge \mathsf{b} \in \mathsf{B}) \rightarrow \exists \mathsf{C}(\mathsf{C} \in \mathcal{F} \wedge \exists \mathsf{x}(\mathsf{x} \in \mathsf{C} \wedge \varphi(\mathsf{x}, \mathsf{a}, \mathsf{b})))).$$

Now let A and B be arbitrary sets. From the last line above, we may conclude that there exists $\mathcal{F}$ such that for all $a \in A$ and all $b \in B$, there exists $C \in \mathcal{F}$ with $(a, b) \in C$. Let $D = \bigcup \mathcal{F}$. Given any $a \in A$ and $b \in B$, we then have $(a, b) \in D$. Now applying Separation to the set D and the formula $\exists \mathsf{a} \exists \mathsf{b}(\mathsf{a} \in \mathsf{A} \wedge \mathsf{b} \in \mathsf{B} \wedge \varphi(\mathsf{x}, \mathsf{a}, \mathsf{b}))$, there is a set E such that for all x, we have $x \in E$ if and only if there exists $a \in A$ and $b \in B$ with $x = (a, b)$. As usual, Extensionality gives uniqueness. □

Now that we have ordered pairs and Cartesian products, we can really make some progress.

Definition 9.1.18. *A* relation *is a set R such that every set $x \in R$ is an ordered pair, i.e. if for every $x \in R$, there exists sets a, b such that $x = (a, b)$.*

Given a relation R, we want to define its domain to be the set of first elements of ordered pairs which are elements of R, and we want to define its range to be the set of second elements of ordered pairs which are elements of R. These are good descriptions which can easily (though not shortly) be turned into formulas, but we need to know that there is some set which contains all of these elements in order to apply Separation. Since the elements of an ordered pair $(a, b) = \{\{a\}, \{a, b\}\}$ are "two deep", a good exercise is to convince yourself that $\bigcup \bigcup R$ will work. This justifies the following definitions.

Definition 9.1.19. *Let R be a relation*

1. *domain(R) is the set of a such that there exists b with $(a, b) \in R$.*
2. *range(R) is the set of b such that there exists a with $(a, b) \in R$.*

We can define the composition of two relations, generalizing the idea of a composition of functions (we talk about this special case below).

Definition 9.1.20. *Let R and S be relations. Let $A =$ domain(R) and $C =$ range(S). We define*

$$S \circ R = \{(a, c) \in A \times C : \text{There exists } b \text{ with } (a, b) \in R \text{ and } (b, c) \in S\}.$$

Definition 9.1.21. *Let R be a relation. We write aRb if $(a, b) \in R$.*

Definition 9.1.22. *Let A be a set. We say that R is a* relation on A *if domain$(R) \subseteq A$ and range$(R) \subseteq A$.*

We define functions in the obvious way.

Definition 9.1.23. *A* function *f is a relation which is such that for all $a \in$ domain(f), there exists a unique $b \in$ range(f) such that $(a, b) \in f$.*

Definition 9.1.24. *Let f be a function. We write $f(a) = b$ if $(a, b) \in f$.*

The definition of function composition is now a special case of the definition of the composition of relations. We do need the following result.

Proposition 9.1.25. *If f and g are both functions, then $g \circ f$ is a function with domain$(g \circ f) \subseteq$ domain(f).*

Definition 9.1.26. *Let f be a function. f is* injective *(or* an injection*) if whenever $f(a_1) = b$ and $f(a_2) = b$, we have $a_1 = a_2$.*

Definition 9.1.27. *Let A and B be sets. We write $f : A \rightarrow B$ to mean that f is a function, domain$(f) = A$ and range$(f) \subseteq B$.*

We are now in a position to define when a function f is surjective and bijective. Notice that surjectivity and bijectivity are not properties of a function itself because these notions depend on a set which you consider to contain range(f). Once we have a fixed such set in mind, however, we can make the definitions.

Definition 9.1.28. *Let A and B be sets, and let $f\colon A \to B$.*

1. *f is* surjective *(or* a surjection*) if range$(f) = B$.*
2. *f is* bijective *(or* a bijection*) if f is injective and surjective.*

Definition 9.1.29. *Let A and B be sets.*

1. *We write $A \preceq B$ to mean that there is an injection $f\colon A \to B$.*
2. *We write $A \approx B$ to mean that there is a bijection $f\colon A \to B$.*

Proposition 9.1.30. *Let A, B, and C be sets. If $A \preceq B$ and $B \preceq C$, then $A \preceq C$.*

Proof. Use the fact that the composition of injective functions is injective. □

Proposition 9.1.31. *Let A, B, and C be sets.*

1. *$A \approx A$.*
2. *If $A \approx B$, then $B \approx A$.*
3. *If $A \approx B$ and $B \approx C$, then $A \approx C$.*

Definition 9.1.32. *Let R be a relation on a set A.*

1. *R is* reflexive on A *if for all $a \in A$, we have aRa.*
2. *R is* symmetric on A *if for all $a, b \in A$, if aRb then bRa.*
3. *R is* asymmetric on A *if for all $a, b \in A$, if aRb then it is not the case that bRa.*
4. *R is* antisymmetric on A *if for all $a, b \in A$, if aRb and bRa, then $a = b$.*
5. *R is* transitive on A *if for all $a, b, c \in A$, if aRb and bRc, then aRc.*
6. *R is* connected on A *if for all $a, b \in A$, either aRb, $a = b$, or bRa.*

Definition 9.1.33. *Let R be a relation on a set A.*

1. *R is a* (strict) partial ordering on A *if R is transitive on A and asymmetric on A.*
2. *R is a* (strict) linear ordering on A *if R is a partial ordering on A and R is connected on A.*
3. *R is a* (strict) well-ordering on A *if R is a linear ordering on A and for every $X \subseteq A$ with $X \neq \emptyset$, there exists $m \in X$ such that for all $x \in X$, either $m = x$ or mRx.*

9.2 The Natural Numbers and Induction

We specifically added the Axiom of Infinity with the hope that it captured the idea of the set of natural numbers. We now show how this axiom, in league with the others, allows us to embed the theory of the natural numbers into set theory. We start by defining the initial natural number and successors of sets.

Definition 9.2.1. $0 = \emptyset$

Definition 9.2.2. *Given a set x, we let $S(x) = x \cup \{x\}$, and we call $S(x)$ the* successor *of x.*

With 0 and the notion of successor, we can then go on to define $1 = S(0)$, $2 = S(1) = S(S(0))$, and continue in this way to define any particular natural number. However, we want to form the set of all natural numbers.

Definition 9.2.3. *A set I is* inductive *if $0 \in I$ and for all $x \in I$, we have $S(x) \in I$.*

The Axiom of Infinity simply asserts the existence of some inductive set J. Intuitively, we have $0 \in J$, $S(0) \in J$, $S(S(0)) \in J$, and so on. However, J may very well contain more than just repeated applications of S to 0. We now use the top-down approach of generation to define the natural numbers (the other two approaches will not work yet because their very definitions rely on the natural numbers).

Proposition 9.2.4. *There is a smallest inductive set. That is, there is an inductive set K such that $K \subseteq I$ for every inductive set I.*

Proof. By the Axiom of Infinity, we may fix an inductive set J. Let $K = \{x \in J : x \in I \text{ for every inductive set } I\}$. Notice that $0 \in K$ because $0 \in I$ for every inductive set I (and so, in particular, $0 \in J$). Suppose that $x \in K$. If I is inductive, then $x \in I$, hence $S(x) \in I$. It follows that $S(x) \in I$ for every inductive set I (and so, in particular, $S(x) \in J$), hence $S(x) \in K$. Therefore, K is inductive. By definition of K, we have $K \subseteq I$ whenever I is inductive. □

By Extensionality, there is a unique smallest inductive set, so this justifies the following definition.

Definition 9.2.5. *We denote the unique smallest inductive set by ω.*

We think that ω captures our intuitive idea of the set of natural numbers, and it is now our goal to show how to prove the basic statements about the natural numbers which are often accepted axiomatically. We first define a relation $<$ on ω. Remember our intuitive idea is that $\in$ captures the order relationship on the natural numbers.

Definition 9.2.6.

1. *We define a relation $<$ on ω by setting $< = \{(n, m) \in \omega \times \omega : n \in m\}$.*
2. *We define a relation $\leq$ on ω by setting $\leq = \{(n, m) \in \omega \times \omega : n < m \text{ or } n = m\}$.*
3. *We define a relation $>$ on ω by setting $> = \{(n, m) \in \omega \times \omega : m < n\}$.*
4. *We define a relation $\geq$ on ω by setting $\geq = \{(n, m) \in \omega \times \omega : n > m \text{ or } n = m\}$.*

Lemma 9.2.7. *There is no $n \in \omega$ with $n < 0$.*

Proof. Since $0 = \emptyset$, there is no set x such that $x \in 0$. Therefore, there is no $n \in \omega$ with $n < 0$. □

Lemma 9.2.8. *Let $m, n \in \omega$ be arbitrary. We have $m < S(n)$ if and only if $m \leq n$.*

Proof. Let $m, n \in \omega$. We then have $S(n) \in \omega$ since ω is inductive, and

$$\begin{aligned} m < S(n) &\Leftrightarrow m \in S(n) \\ &\Leftrightarrow m \in n \cup \{n\} \\ &\Leftrightarrow \text{Either } m \in n \text{ or } m \in \{n\} \\ &\Leftrightarrow \text{Either } m < n \text{ or } m = n \\ &\Leftrightarrow m \leq n. \end{aligned}$$

This proves the lemma. □

Our primary objective is to show that $<$ is a well-ordering on ω. Due to the nature of the definition of ω, it seems that only way to prove nontrivial results about ω is "by induction". We state the Step Induction Principle in two forms. The first is much cleaner and seemingly more powerful (because it immediately implies the second and we can quantify over sets but not over formulas), but the second is how one often thinks about induction in practice by using "properties" of natural numbers, and will be the only form that we can generalize to the collection of all ordinals.

Proposition 9.2.9 (Step Induction Principle on ω)**.**

1. *Suppose that X is a set, $0 \in X$, and for all $n \in \omega$, if $n \in X$ then $S(n) \in X$. We then have $\omega \subseteq X$.*
2. *For any formula $\varphi(\mathsf{n}, \vec{\mathsf{p}})$, the sentence*

$$\forall \vec{\mathsf{p}}((\varphi(0, \vec{\mathsf{p}}) \wedge (\forall \mathsf{n} \in \omega)(\varphi(\mathsf{n}, \vec{\mathsf{p}}) \to \varphi(S(\mathsf{n}), \vec{\mathsf{p}}))) \to (\forall \mathsf{n} \in \omega)\varphi(\mathsf{n}, \vec{\mathsf{p}}))$$

is in ZFC, where $\varphi(0, \vec{\mathsf{p}})$ is shorthand for the formula

$$\exists \mathsf{x}(\forall \mathsf{y}(\neg(\mathsf{y} \in \mathsf{x})) \wedge \varphi(\mathsf{x}, \vec{\mathsf{p}})),$$

and $\varphi(S(\mathsf{n}), \vec{\mathsf{p}})$ is shorthand for the formula

$$\exists \mathsf{x}(\forall \mathsf{y}(\mathsf{y} \in \mathsf{x} \leftrightarrow (\mathsf{y} \in \mathsf{n} \vee \mathsf{y} = \mathsf{n})) \wedge \varphi(\mathsf{x}, \vec{\mathsf{p}})).$$

Proof.

1. Let $Y = X \cap \omega$. Notice first that $0 \in Y$. Suppose now that $n \in Y = X \cap \omega$. We then have $n \in \omega$ and $n \in X$, so $S(n) \in \omega$ (because ω is inductive), and $S(n) \in X$ by assumption. Hence, $S(n) \in Y$. Therefore, Y is inductive, so we may conclude that $\omega \subseteq Y$. It follows that $\omega \subseteq X$.
2. Let $\vec{q}$ be an arbitrary sequence of sets, and suppose $\varphi(0, \vec{q})$ and $(\forall n \in \omega)(\varphi(n, \vec{q}) \to \varphi(S(n), \vec{q}))$. Let $X = \{n \in \omega : \varphi(n, \vec{q})\}$, which exists by Separation. Notice that $0 \in X$ and for all $n \in \omega$, if $n \in X$ then $S(n) \in X$ by assumption. It follows from part 1 that $\omega \subseteq X$. Therefore, we have $(\forall n \in \omega)\varphi(n, \vec{q})$.

□

With the Step Induction Principle in hand, we can begin to prove the basic facts about the natural numbers. Our goal is to prove that $<$ is a well-ordering on ω, but it will take some time to get there. We first give a very simple inductive proof. For this proof only, we will give careful arguments using both versions of Step Induction to show how a usual induction proof can be formalized in either way.

Lemma 9.2.10. *For all $n \in \omega$, we have $0 \leq n$.*

Proof. The following two proofs correspond to the above two versions of the Induction Principle.

1. Let $X = \{n \in \omega : 0 \leq n\}$, and notice that $0 \in X$. Suppose now that $n \in X$. We then have $n \in \omega$ and $0 \leq n$, hence $0 < S(n)$ by Lemma 9.2.8, so $S(n) \in X$. Thus, by Step Induction, we have $\omega \subseteq X$. Therefore, for all $n \in \omega$, we have $0 \leq n$.

2. Let $\varphi(n)$ be the formula "$0 \leq n$". We clearly have $\varphi(0)$ because $0 = 0$. Suppose now that $n \in \omega$ and $\varphi(n)$. We then have $0 \leq n$, hence $0 < S(n)$ by Lemma 9.2.8. It follows that $\varphi(S(n))$. Therefore, by Step Induction, we have $0 \leq n$ for all $n \in \omega$.

□

We give a few more careful inductive proof using the second version of the Induction Principle to illustrate how parameters can be used. Afterwards, our later inductive proofs will be given in a more natural relaxed style.

Our relation $<$ is given by $\in$, but it is only defined on elements of ω. We thus need the following proposition which says that every element of a natural number is a natural number.

Proposition 9.2.11. *Suppose that $n \in \omega$ and $m \in n$. We then have $m \in \omega$.*

Proof. The proof is "by induction on n"; that is, we hold m fixed by treating it as a parameter. Thus, let $m \in \omega$ be arbitrary, and let $X = \{n \in \omega : m \in n \to m \in \omega\}$. In other words, we have $X = \{n \in \omega : m \notin n \text{ or } m \in \omega\}$. Notice that $0 \in X$ because $m \notin 0 = \emptyset$. Suppose now that $n \in X$. We show that $S(n) \in X$. Suppose that $m \in S(n) = n \cup \{n\}$ (otherwise we trivially have $m \in X$). We then know that either $m \in n$, in which case $m \in \omega$ by induction (i.e. because $n \in X$), or $m = n$, in which case we clearly have $m \in \omega$. It follows that $S(n) \in X$. Therefore, by Step Induction, we may conclude that $X = \omega$. Since $m \in \omega$ was arbitrary, the result follows. □

Proposition 9.2.12. *$<$ is transitive on ω.*

Proof. We prove the result by induction on n. Let $k, m \in \omega$ be arbitrary, and let

$$X = \{n \in \omega : (k < m \wedge m < n) \to k < n\}.$$

We then have that $0 \in X$ vacuously because we do not have $m < 0$ by Lemma 9.2.7. Suppose now that $n \in X$. We show that $S(n) \in X$. Suppose that $k < m$ and $m < S(n)$ (if not, then $S(n) \in X$ vacuously). By Lemma 9.2.8, we have $m \leq n$, hence either $m < n$ or $m = n$. If $m < n$, then $k < n$ because $n \in X$. If $m = n$, then $k < n$ because $k < m$. Therefore, in either case, we have $k < n$, and hence $k < S(n)$ by Lemma 9.2.8. It follows that $S(n) \in X$. Thus, by Step Induction, we may conclude that $X = \omega$. Since $k, m \in \omega$ were arbitrary, the result follows. □

Lemma 9.2.13. *Let $m, n \in \omega$. We have $S(m) \leq n$ if and only if $m < n$.*

Proof. Suppose first that $m, n \in \omega$ and $S(m) \leq n$.

- *Case 1:* Suppose that $S(m) = n$. We have $m < S(m)$ by Lemma 9.2.8, hence $m < n$.
- *Case 2:* Suppose that $S(m) < n$. We have $m < S(m)$ by Lemma 9.2.8, hence $m < n$ by Proposition 9.2.12.

Therefore, for all $n, m \in \omega$, if $S(m) \leq n$, then $m < n$.

We prove the converse statement that for all $m, n \in \omega$, if $m < n$, then $S(m) \leq n$ by induction on n. Let $m \in \omega$ be arbitrary, and let $X = \{n \in \omega : m < n \to S(m) \leq n\}$. We have $0 \in X$ vacuously because we do not have $m < 0$ by Lemma 9.2.7. Suppose now that $n \in X$. We show that $S(n) \in X$. Suppose that $m < S(n)$ (otherwise $S(n) \in X$ vacuously). By Lemma 9.2.8, we have $m \leq n$.

- *Case 1:* Suppose that $m = n$. We then have $S(m) = S(n)$, hence $S(n) \in X$.

- *Case 2:* Suppose that $m < n$. Since $n \in X$, we have $S(m) \leq n$. By Lemma 9.2.8, we know that $n < S(n)$. If $S(m) = n$, this immediately gives $S(m) < S(n)$, while if $S(m) < n$, we may conclude that $S(m) < S(n)$ by Proposition 9.2.12. Hence, we have $S(n) \in X$.

Thus, by Step Induction, we may conclude that $X = \omega$. Since $m \in \omega$ was arbitrary, the result follows. □

Lemma 9.2.14. *There is no $n \in \omega$ with $n < n$.*

Proof. This follows immediately from the Axiom of Foundation, but we prove it without that assumption. Let $X = \{n \in \omega : \neg(n < n)\}$. We have that $0 \in X$ by Lemma 9.2.7. Suppose that $n \in X$. We prove that $S(n) \in X$ by supposing that $S(n) < S(n)$ and deriving a contradiction. Suppose then that $S(n) < S(n)$. By Lemma 9.2.8, we have $S(n) \leq n$, hence either $S(n) = n$ or $S(n) < n$. Also by Lemma 9.2.8, we have $n < S(n)$. Therefore, if $S(n) = n$, then $n < n$, and if $S(n) < n$, then $n < n$ by Proposition 9.2.12 (since $n < S(n)$ and $S(n) < n$), a contradiction. It follows that $S(n) \in X$. Therefore, there is no $n \in \omega$ with $n < n$. □

Proposition 9.2.15. *$<$ is asymmetric on ω.*

Proof. Suppose that $n, m \in \omega$, $n < m$, and $m < n$. By Proposition 9.2.12, it follows that $n < n$, contradicting Lemma 9.2.14 □

Proposition 9.2.16. *$<$ is connected on ω.*

Proof. We prove that for all $m, n \in \omega$, either $m < n$, $m = n$, or $n < m$ by induction on n. Let $m \in \omega$ be arbitrary, and let $X = \{n \in \omega : (m < n) \vee (m = n) \vee (n < m)\}$. We have $0 \leq m$ by Lemma 9.2.10, hence either $m = 0$ or $0 < m$, and so $0 \in X$. Suppose then that $n \in X$, so that either $m < n$, $m = n$, or $n < m$.

- *Case 1:* Suppose that $m < n$. Since $n < S(n)$ by Lemma 9.2.8, we have $m < S(n)$ by Proposition 9.2.12.
- *Case 2:* Suppose that $m = n$. Since $n < S(n)$ by Lemma 9.2.8, it follows that $m < S(n)$.
- *Case 3:* Suppose that $n < m$. We have $S(n) \leq m$ by Lemma 9.2.13. Hence, either $m = S(n)$ or $S(n) < m$.

Therefore, in all cases, either $m < S(n)$, $m = S(n)$, or $S(n) < m$, so $S(n) \in X$. The result follows by induction. □

In order to finish off the proof that $<$ is a well-ordering on ω, we need a new version of induction. You may have heard it referred to as "Strong Induction".

Proposition 9.2.17 (Induction Principle on ω)**.**

1. *Suppose that X is set and for all $n \in \omega$, if $m \in X$ for all $m < n$, then $n \in X$. We then have $\omega \subseteq X$.*
2. *For any formula $\varphi(\mathsf{n}, \vec{\mathsf{p}})$, we have the sentence*

$$\forall \vec{\mathsf{p}}((\forall \mathsf{n} \in \omega)((\forall \mathsf{m} < \mathsf{n})\varphi(\mathsf{m}, \vec{\mathsf{p}}) \to \varphi(\mathsf{n}, \vec{\mathsf{p}})) \to (\forall \mathsf{n} \in \omega)\varphi(\mathsf{n}, \vec{\mathsf{p}}))$$

Proof.

1. Let $Y = \{n \in \omega : (\forall m < n)(m \in X)\}$. Notice that $Y \subseteq \omega$ and $0 \in Y$ because there is no $m \in \omega$ with $m < 0$ by Lemma 9.2.7. Suppose that $n \in Y$. We show that $S(n) \in Y$. Suppose that $m < S(n)$. By Lemma 9.2.8, we have $m \leq n$, hence either $m < n$ or $m = n$. If $m < n$, then $m \in X$ because $n \in Y$. For the case $m = n$, notice that $n \in X$ by assumption (because $m \in X$ for all $m < n$). Therefore, $S(n) \in Y$. By Step Induction, it follows that $\omega \subseteq Y$.

   Now let $n \in \omega$ be arbitrary. We have $n \in \omega$, hence $S(n) \in \omega$ because ω is inductive, so $S(n) \in Y$. Since $n < S(n)$ by Lemma 9.2.8, it follows that $n \in X$. Therefore, $\omega \subseteq X$.

2. This follows from part 1 using Separation. Fix sets $\vec{q}$, and suppose that

$$(\forall n \in \omega)((\forall m < n)\varphi(m, \vec{q}) \to \varphi(n, \vec{q}))$$

Let $X = \{n \in \omega : \varphi(n, \vec{q})\}$. Suppose that $n \in \omega$ and $m \in X$ for all $m < n$. We then have $(\forall m < n)\varphi(m, \vec{q})$, hence $\varphi(n, \vec{q})$ by assumption, so $n \in X$. It follows from part 1 that $\omega \subseteq X$. Therefore, we have $(\forall n \in \omega)\varphi(n, \vec{q})$.

□

It is possible to give a proof of part 2 which makes use of part 2 of the Step Induction Principle, thus avoiding the detour through sets and using only formulas. This proof simply mimics how we obtained part 1 above, but uses formulas everywhere instead of working with sets. Although it is not nearly as clean, when we treat ordinals, there will times when we need to argue at the level of formulas.

Theorem 9.2.18. *$<$ is a well-ordering on ω*

Proof. By Proposition 9.2.12, Proposition 9.2.15, and Proposition 9.2.16, it follows that $<$ is a linear ordering on ω. Suppose then that $Z \subseteq \omega$ and there is no $n \in Z$ such that for all $m \in Z$, either $n = m$ or $n < m$. We show that $Z = \emptyset$. Notice that for every $n \in Z$, there exists $m \in Z$ with $m < n$ by Proposition 9.2.12.

Let $Y = \omega \backslash Z$. We show that $Y = \omega$ using the Induction Principle. Let $n \in \omega$ be arbitrary with the property that $m \in Y$ for all $m < n$. We then have that $m \notin Z$ for all $m < n$. Therefore, by the last sentence of the previous paragraph, we must have that $n \notin Z$, and so $n \in Y$. By the Induction Principle, we have that $Y = \omega$, and hence $Z = \emptyset$.

Therefore, if $Z \subseteq \omega$ and $Z \neq \emptyset$, there exists $n \in X$ such that for all $m \in Z$, either $n = m$ or $n < m$. It follows that $<$ is a well-ordering on ω. □

9.3 Sets and Classes

We know from Proposition 9.1.4 that there is no set u such that $a \in u$ for all sets a. Thus, our theory forbids us from placing every set into one universal set which we can then play with and manipulate. However, this formal impossibility within our theory does not prevent us from thinking about or referring to the "collection" of all sets or other "collections" which are too "large" to form into a set. After all, our universal quantifiers do indeed range over the "collection" of all sets. Also, if we are arguing semantically, then given a model $\mathcal{M}$ of ZFC, we may "externally" work with the power set of M.

We want to be able to reason about such "collections" of sets in a natural manner within our theory without violating our theory. We will call such "collections" *classes* to distinguish them from sets. The idea is to recall that any first-order theory can say things about certain subsets of every model: the definable subsets. In our case, a formula $\varphi(\mathsf{x})$ is implicitly defining a certain collection of sets. Perhaps this collection is too "large" to put together into a set inside the model, but we may nevertheless use the formula in various ways within our theory. For example, for any formulas $\varphi(\mathsf{x})$ and $\psi(\mathsf{x})$, the sentence $\forall \mathsf{x}(\varphi(\mathsf{x}) \to \psi(\mathsf{x}))$ says that every set which satisfies φ also satisfies ψ. If there exist sets C and D such that $\forall \mathsf{x}(\varphi(\mathsf{x}) \to \mathsf{x} \in C)$ and $\forall \mathsf{x}(\psi(\mathsf{x}) \to \mathsf{x} \in D)$, then we can use Separation to form the sets $A = \{x \in C : \varphi(x)\}$ and $B = \{x \in D : \psi(x)\}$, in which case the sentence $\forall \mathsf{x}(\varphi(\mathsf{x}) \to \psi(\mathsf{x}))$ simply asserts that $A \subseteq B$. However, even if we can't form these sets (intuitively because $\{x : \varphi(x)\}$ and $\{x : \psi(x)\}$ are too "large" to be sets), the sentence is expressing the same underlying idea. Allowing the possibility of parameters, this motivates the following "internal"' definition.

Definition 9.3.1. *A* class ***C*** *is a formula $\varphi(\mathsf{x}, \vec{\mathsf{p}})$.*

Our course, this isn't a very good way to think about classes. Externally, a class is simply a definable set (with the possibility of parameters). The idea is that once we fix sets $\vec{q}$ to fill in for the position of the parameters, the formula describes the collection of those sets a such that $\varphi(a, \vec{q})$. The first class to consider is the class of all sets, which we denote by $\mathbf{V}$. Formally, we define $\mathbf{V}$ to be the formula $\mathsf{x} = \mathsf{x}$, but we will content ourselves with defining classes in the following more informal "external" style.

Definition 9.3.2. *$\boldsymbol{V}$ is the class of all sets.*

Here's a more interesting illustration of how classes can be used and why we want to consider them. Let $\mathbf{C}_R$ be the class of all relations and let $\mathbf{C}_F$ be the class of all functions. More formally, $\mathbf{C}_R$ is the formula $\varphi_R(\mathsf{x})$ given by

$$\forall \mathsf{y}(\mathsf{y} \in \mathsf{x} \to \exists \mathsf{a} \exists \mathsf{b}(\mathsf{y} = (\mathsf{a}, \mathsf{b})))$$

while $\mathbf{C}_F$ is the formula $\varphi_F(x)$ given by

$$\forall \mathsf{y}(\mathsf{y} \in \mathsf{x} \to \exists \mathsf{a} \exists \mathsf{b}(\mathsf{y} = (\mathsf{a}, \mathsf{b}))) \wedge \forall \mathsf{a} \forall \mathsf{b}_1 \forall \mathsf{b}_2(((\mathsf{a}, \mathsf{b}_1) \in \mathsf{x} \wedge (\mathsf{a}, \mathsf{b}_2) \in \mathsf{x}) \to \mathsf{b}_1 = \mathsf{b}_2)$$

With this shorthand in place, we can write things like $\mathbf{C}_F \subseteq \mathbf{C}_R$ to stand for the provable sentence $\forall \mathsf{x}(\varphi_F(\mathsf{x}) \to \varphi_R(\mathsf{x}))$. Thus, by using the language of classes, we can express complicated formulas in a simplified, more suggestive, fashion. Of course, there's no real need to introduce classes because we could always just refer to the formulas, but it is psychologically easier to think of a class as some kind of ultra-set which our theory is able to handle, even if we are limited in what we can do with classes.

With the ability to refer to classes, why deal with sets at all? The answer is that classes are much less versatile than sets. For example, if $\mathbf{C}$ and $\mathbf{D}$ are classes, it makes no sense to write $\mathbf{C} \in \mathbf{D}$ because this doesn't correspond to a formula built from the implicit formulas giving $\mathbf{C}$ and $\mathbf{D}$. This inability corresponds to the intuition that classes are too "large" to collect together into a set and then put into other collections. Hence, asking whether $\mathbf{V} \in \mathbf{V}$ is meaningless. Also, since classes are given by formulas, we are restricted to referring only to "definable" collections. Thus, there is no way to talk about or quantify over all "collections" of sets (something that is meaningless internally). However, there are many operation which do make sense on classes.

For instance, suppose that $\mathbf{R}$ is a class of ordered pairs (with parameters $\vec{p}$). That is, $\mathbf{R}$ is a formula $\varphi(\mathsf{x}, \vec{\mathsf{p}})$ such that the formula $\forall \mathsf{x}(\varphi(\mathsf{x}, \vec{\mathsf{p}}) \to \exists \mathsf{a} \exists \mathsf{b}(\mathsf{x} = (\mathsf{a}, \mathsf{b})))$ is provable. We think of $\mathbf{R}$ as a *class relation*. Using suggestive notation, we can then go on to define domain($\mathbf{R}$) to be the class consisting of those sets a such that there exists a set b with $(a, b) \in \mathbf{R}$. To be precise, domain($\mathbf{R}$) is the class which is the formula $\psi(\mathsf{a}, \vec{\mathsf{p}})$ given by $\exists \mathsf{x} \exists \mathsf{b}(\mathsf{x} = (\mathsf{a}, \mathsf{b}) \wedge \varphi(\mathsf{x}, \vec{\mathsf{p}}))$. Thus, we can think of domain$(\cdot)$ as a operation on classes (given any formula $\varphi(\mathsf{x}, \vec{\mathsf{p}})$ which is a class relation, applying domain$(\cdot)$ results in the class given by the formula $\exists \mathsf{x} \exists \mathsf{b}(\mathsf{x} = (\mathsf{a}, \mathsf{b}) \wedge \varphi(\mathsf{x}, \vec{\mathsf{p}}))$).

Similarly, we can talk about *class functions*. We can even use notation like $\mathbf{F} : \mathbf{V} \to \mathbf{V}$ to mean that $\mathbf{F}$ is a class function with domain($\mathbf{F}$) $= \mathbf{V}$. Again, each of these expressions could have been written out as formulas in our language, but the notation is so suggestive that it's clear how to do this without actually having to do it. An example of a general class function is $\mathbf{U} : \mathbf{V} \times \mathbf{V} \to \mathbf{V}$ given by $\mathbf{U}(a, b) = a \cup b$. Convince yourself how to write $\mathbf{U}$ as a formula.

We can not quantify over classes within our theory in the same way that we can quantify over sets because there is no way to quantify over the formulas of set theory within set theory. However, we can, at the price of considering one "theorem" as infinitely many (one for each formula), make sense of a theorem which does universally quantify over classes. For example, consider the following.

Proposition 9.3.3. *Suppose that $\boldsymbol{C}$ is a class, $0 \in \boldsymbol{C}$, and for all $n \in \omega$, if $n \in \boldsymbol{C}$ then $S(n) \in \boldsymbol{C}$. We then have $\omega \subseteq \boldsymbol{C}$.*

This proposition is what is obtained from the first version of Step Induction on ω by replacing the set X with the class $\mathbf{C}$. Although the set version can be written as one sentence which is provable in ZFC, this

version can not because we can't quantify over classes in the the theory. Unwrapping this proposition into formulas, it says that for every formula $\varphi(\mathsf{x}, \vec{\mathsf{p}})$, if we can prove $\varphi(0, \vec{\mathsf{p}})$ and $(\forall \mathsf{n} \in \omega)(\varphi(\mathsf{n}, \vec{\mathsf{p}}) \to \varphi(S(\mathsf{n}), \vec{\mathsf{p}}))$, then we can prove $(\forall \mathsf{n} \in \omega)\varphi(\mathsf{n}, \vec{\mathsf{p}})$. That is, for each formula $\varphi(\mathsf{x}, \vec{\mathsf{p}})$, we can prove the sentence

$$\forall \vec{\mathsf{p}}((\varphi(0, \vec{\mathsf{p}}) \wedge (\forall \mathsf{n} \in \omega)(\varphi(\mathsf{n}, \vec{\mathsf{p}}) \to \varphi(S(\mathsf{n}), \vec{\mathsf{p}}))) \to (\forall \mathsf{n} \in \omega)\varphi(\mathsf{n}, \vec{\mathsf{p}})).$$

Thus, the class version is simply a neater way of writing the second version of Step Induction on ω which masks the fact that the quantification over classes requires us to write it as infinitely many different propositions (one for each formula $\varphi(\mathsf{x}, \vec{\mathsf{p}})$) in our theory.

Every set can be viewed as a class by making use of the class $\mathbf{M}$ given by the formula $\mathsf{x} \in \mathsf{p}$. That is, once we fix a set p, the class $\mathsf{x} \in \mathsf{p}$ describes exactly the elements of p. For example, using $\mathbf{M}$ in class version of Step Induction on ω, we see that the following sentence is provable:

$$\forall \mathsf{p}((0 \in \mathsf{p} \wedge (\forall \mathsf{n} \in \omega)(\mathsf{n} \in \mathsf{p} \to S(\mathsf{n}) \in \mathsf{p})) \to (\forall \mathsf{n} \in \omega)(\mathsf{n} \in \mathsf{p})).$$

Notice that this is exactly the set version of Step Induction on ω.

On the other hand, not every class can be viewed as a set (look at $\mathbf{V}$, for example). Let $\mathbf{C}$ be a class. We say that $\mathbf{C}$ is a set if there exists a set A such that for all x, we have $x \in \mathbf{C}$ if and only if $x \in A$. At the level of formulas, this means that if $\mathbf{C}$ is given by the formula $\varphi(\mathsf{x}, \vec{\mathsf{p}})$, then we can prove the formula $\exists \mathsf{A} \forall \mathsf{x}(\varphi(\mathsf{x}, \vec{\mathsf{p}}) \leftrightarrow \mathsf{x} \in \mathsf{A})$. By Separation, this is equivalent to saying that there is a set B such that for all x, if $x \in \mathbf{C}$ then $x \in B$ (i.e. we can prove the formula $\exists \mathsf{B} \forall \mathsf{x}(\varphi(\mathsf{x}, \vec{\mathsf{p}}) \to \mathsf{x} \in \mathsf{B})$).

Definition 9.3.4. *Let $\boldsymbol{C}$ be a class defined by a formula $\varphi(\mathsf{x}, \vec{\mathsf{p}})$. We say that $\boldsymbol{C}$ is a* proper class *if $\boldsymbol{C}$ is not a set, i.e. if we can prove $\neg(\exists \mathsf{A} \forall \mathsf{x}(\varphi(\mathsf{x}, \vec{\mathsf{p}}) \leftrightarrow \mathsf{x} \in \mathsf{A}))$.*

For example, $\mathbf{V}$ is a proper class. The following proposition will be helpful to us when we discuss transfinite constructions. Intuitively, it says that proper classes are too large to embedded into any set.

Proposition 9.3.5. *Let $\boldsymbol{C}$ be a proper class and let A be a set. There is no injective class function $\boldsymbol{F} : \boldsymbol{C} \to A$.*

Proof. Suppose that $\mathbf{F} : \mathbf{C} \to A$ is an injective class function. Let $B = \{a \in A : \exists c(c \in \mathbf{C} \wedge \mathbf{F}(c) = a)\}$ and notice that B is a set by Separation (recall that $\mathbf{C}$ and $\mathbf{F}$ are given by formulas). Since for each $b \in B$, there is a unique $c \in \mathbf{C}$ with $\mathbf{F}(c) = b$ (using the fact that $\mathbf{F}$ is injective), we may use Collection and Separation to conclude that $\mathbf{C}$ is a set, contradicting the fact that $\mathbf{C}$ is a proper class. □

We end this section by seeing how to simply restate the Axiom of Separation and the Axiom of Collection in the language of classes.

Axiom of Separation: Every subclass of a set is a set.

Axiom of Collection: If $\mathbf{F}$ is a class function and A is a set, then there is a set containing the image of A under $\mathbf{F}$.

9.4 Finite Sets and Finite Powers

Definition 9.4.1. *Let A be a set. A is* finite *if there exists $n \in \omega$ such that $A \approx n$. If A is not finite, we say that A is* infinite*.*

Proposition 9.4.2. *Suppose that $n \in \omega$. Every injective $f : n \to n$ is bijective.*

Proof. The proof is by induction on $n \in \omega$. Suppose first that $n = 0$ and $f\colon 0 \to 0$ is injective. We then have $f = \emptyset$, so f is trivially bijective. Assume now that the result holds for n, i.e. assume that every injective $f\colon n \to n$ is bijective. Let $f\colon S(n) \to S(n)$ be an arbitrary injective function. We then have $f(n) \leq n$, and we consider two cases.

- *Case 1:* Suppose that $f(n) = n$. Since f is injective, we have $f(m) \neq n$ for every $m < n$, hence $f(m) < n$ for every $m < n$ (because $f(m) < S(n)$ for every $m < n$). It follows that $f \restriction n\colon n \to n$. Notice that $f \restriction n\colon n \to n$ is injective because f is injective, hence $f \restriction n$ is bijective by induction. Therefore, $\operatorname{range}(f \restriction n) = n$, and hence $\operatorname{range}(f) = S(n)$ (because $f(n) = n$). It follows that f is surjective, so f is bijective.

- *Case 2:* Suppose that $f(n) < n$. We first claim that $n \in \operatorname{range}(f)$. Suppose instead that $n \notin \operatorname{range}(f)$. Notice that $f \restriction n\colon n \to n$ is injective because f is injective, hence $f \restriction n$ is bijective by induction. Therefore, $f(n) \in \operatorname{range}(f \restriction n)$ (because $f(n) < n$), so there exists $\ell < n$ with $f(\ell) = f(n)$, contrary to the fact that f is injective. It follows that $n \in \operatorname{range}(f)$.

  Fix $k < n$ with $f(k) = n$. Define a function $g\colon n \to n$ by

$$g(m) = \begin{cases} f(m) & \text{if } m \neq k \\ f(n) & \text{if } m = k. \end{cases}$$

  Notice that if $m_1, m_2 < n$ with $m_1 \neq m_2$ and $m_1, m_2 \neq k$, then $g(m_1) \neq g(m_2)$ since $f(m_1) \neq f(m_2)$ (because f is injective). Also, if $m < n$ with $m \neq k$, then $g(m) \neq g(k)$ since $f(m) \neq f(n)$ (again because f is injective). It follows that $g\colon n \to n$ is injective, hence bijective by induction. From this we can conclude that $\operatorname{range}(f) = S(n)$ as follows. Notice that $f(n) \in \operatorname{range}(f)$ trivially, and $n \in \operatorname{range}(f)$ because $f(k) = n$. Suppose that $\ell < n$ with $\ell \neq f(n)$. Since $g\colon n \to n$ is bijective, there exists a unique $m < n$ with $g(m) = \ell$. Since $\ell \neq f(n)$, we have $m \neq k$, hence $f(m) = g(m) = \ell$, so $\ell \in \operatorname{range}(f)$. Therefore, $\operatorname{range}(f) = S(n)$, and hence f is bijective.

□

Corollary 9.4.3 (Pigeonhole Principle)**.** *If $n, m \in \omega$ and $m > n$, then $m \not\preceq n$.*

Proof. Suppose that $f\colon m \to n$ is injective. It then follows that $f \restriction n\colon n \to n$ is injective, hence $f \restriction n$ is bijective by Proposition 9.4.2. Therefore, since $f(n) \in n$, it follows that there exists $k < n$ with $f(k) = f(n)$, contradicting the fact that f is injective. Hence, $m \not\preceq n$. □

Corollary 9.4.4. *If $m, n \in \omega$ and $m \approx n$, then $m = n$.*

Proof. Suppose that $m \neq n$ so that either $m > n$ or $m < n$. If $m > n$, then $m \not\preceq n$ by the Pigeonhole Principle, so $m \not\approx n$. If $m < n$, then $n \not\preceq m$ by the Pigeonhole Principle, so $n \not\approx m$ and hence $m \not\approx n$. □

Corollary 9.4.5. *If A is finite, there exists a unique $n \in \omega$ such that $A \approx n$.*

Definition 9.4.6. *If A is finite, the unique $n \in \omega$ such that $A \approx n$ is called the* cardinality *of A and is denoted by $|A|$.*

Proposition 9.4.7. *Let A be a nonempty set and let $n \in \omega$. The following are equivalent:*

1. *$A \preceq n$.*
2. *There exists a surjection $g\colon n \to A$.*
3. *A is finite and $|A| \leq n$.*

Proof. We prove four implications.

- 1 implies 2: Suppose that $A \preceq n$ and fix an injection $f\colon A \to n$. Fix an element $b \in A$ (which exists since $A \neq \emptyset$). Define a set g by letting

$$g = \{(m, a) \in n \times A : f(a) = m\} \cup \{(m, a) \in n \times A : m \notin \text{range}(f) \text{ and } a = b\}.$$

  Notice that g is a function because f is injective. Furthermore, we have that $\text{domain}(g) = n$ and $\text{range}(g) = A$, so the function $g\colon n \to A$ is surjective.

- 2 implies 1: Suppose that $g\colon n \to A$ is a surjection. Define a set f by letting

$$f = \{(a, m) \in A \times n : g(m) = a \text{ and } g(k) \neq a \text{ for all } k < m\}.$$

  Notice that f is a function because $<$ is connected on ω by Proposition 9.2.16. Furthermore, using the assumption that g is a surjection together with the fact that $<$ is a well-ordering on ω, it follows that $\text{domain}(f) = A$ so $f\colon A \to n$. Finally, we have that f is injective because g is a function.

- 1 implies 3: Suppose that $A \preceq n$. Since $<$ is a well-ordering on ω, we may let m be the least element of ω such that $A \preceq m$. Notice that $m \leq n$. By definition of $A \preceq m$, we can fix an injection $g\colon A \to m$. We claim that g is also surjective. Suppose not, and fix $\ell < m$ such that $\ell \notin \text{range}(g)$. Notice that $m \neq 0$ because A is nonempty, so we may fix $k \in \omega$ with $m = S(k)$. If $\ell = k$, then we can view g as an injective function $g\colon A \to k$, contradicting our choice of m as least. Otherwise, we have $\ell < k$, and then the function $h\colon A \to k$ defined by

$$h(a) = \begin{cases} g(a) & \text{if } g(a) \neq k \\ \ell & \text{otherwise} \end{cases}$$

  would be injective, a contradiction. Therefore, g is surjective, hence bijective, and so $|A| = m \leq n$.

- 3 implies 1: Suppose that A is finite and $|A| \leq n$. Let $m = |A| \leq n$ and fix a bijection $f\colon A \to m$. We then have that $f\colon A \to n$ is an injection, so $A \preceq n$.

□

Corollary 9.4.8. *Suppose that $n \in \omega$. Every surjective $g\colon n \to n$ is bijective.*

Proof. Suppose that $g\colon n \to n$ is surjective. Let

$$f = \{(a, m) \in n \times n : g(m) = a \text{ and } g(k) \neq a \text{ for all } k < m\}$$

and notice that $f\colon n \to n$ is an injective function by the proof of "2 implies 1" above. By Proposition 9.4.2, we know that f is bijective. Now let $k, m \in n$ be arbitrary with $g(k) = g(m)$.

- *Case 1:* Suppose that $k < m$. Since f is bijective, we can fix $b \in n$ with $f(b) = m$. We then have $(b, m) \in f$, so by definition, we must have $g(m) = b$ and $g(k) \neq b$. However, this is a contradiction because $g(m) = g(k)$.

- *Case 2:* Suppose that $m < k$. Since f is bijective, we can fix $b \in n$ with $f(b) = k$. We then have $(b, k) \in f$, so by definition, we must have $g(k) = b$ and $g(m) \neq b$. However, this is a contradiction because $g(k) = g(m)$.

Since $<$ is connected on ω, the only possibility is that $k = m$. Therefore, g is injective, and hence bijective. □

It is possible to use ordered pairs to define ordered triples, ordered quadruples, and so on. For example, we could define the ordered triple (a, b, c) to be $((a, b), c)$. However, with the basic properties of ω in hand, we can give a much more elegant definition.

Proposition 9.4.9. *Let A be a set. For all $n \in \omega$, there is a unique set, denoted by A^n, such that for all f, we have $f \in A^n$ if and only if $f : n \to A$.*

Proof. Let A be an arbitrary set. As usual, uniqueness follows from Extensionality, so we need only prove existence. The proof is by induction on n. Suppose that $n = 0$. Since for all f, we have $f : 0 \to A$ if and only if $f = \emptyset$, we may take $A^0 = \{\emptyset\}$. Suppose that the statement is true for a given $n \in \omega$, i.e. there exists a set A^n such that for all f, we have $f \in A^n$ if and only if $f : n \to A$. We prove it for $S(n)$.

Let $a \in A$ be arbitrary. Notice that for each $f \in A^n$, there is a unique function $f_a : S(n) \to A$ such that $f_a(m) = f(m)$ for all $m < n$ and $f_a(n) = a$ (let $f_a = f \cup \{(n, a)\}$ and use Lemma 9.2.8). Therefore, by Collection (since A^n is a set), Separation, and Extensionality, there is a unique set C_a such that for all g, we have $g \in C_a$ if and only if $g = f_a$ for some $a \in A$. Notice that for every $g : S(n) \to A$ with $g(n) = a$, there is an $f : n \to A$ such that $g = f_a$ (let $f = g \backslash \{(n, a)\}$). Therefore, for every g, we have $g \in C_a$ if and only if $g : S(n) \to A$ and $g(n) = a$.

By Collection (since A is a set), Separation, and Extensionality again, there is a set $\mathcal{F}$ such that for all D, we have $D \in \mathcal{F}$ if and only if there exists $a \in A$ with $D = C_a$. Notice that for all sets g, we have $g \in \bigcup \mathcal{F}$ if and only if there exists $a \in A$ with $g \in C_a$. Let $A^{S(n)} = \bigcup \mathcal{F}$. We then have $g \in A^{S(n)}$ if and only $g : S(n) \to A$. □

Proposition 9.4.10. *Let A be a set. There is a unique set, denoted by $A^{<\omega}$, such that for all f, we have $f \in A^{<\omega}$ if and only if $f \in A^n$ for some $n \in \omega$.*

Proof. By Collection (since ω is a set), Separation, and Extensionality, there is a unique set $\mathcal{F}$ such that for all D, we have $D \in \mathcal{F}$ if and only if there exists $n \in \omega$ with $D = A^n$. Let $A^{<\omega} = \bigcup F$. For every f, we then have $f \in A^{<\omega}$ if and only if $f \in A^n$ for some $n \in \omega$. □

9.5 Definitions by Recursion

Theorem 9.5.1 (Step Recursive Definitions on ω - Set Form)**.** *Let A be a set, let $b \in A$, and let $g : \omega \times A \to A$. There exists a unique function $f : \omega \to A$ such that $f(0) = b$ and $f(S(n)) = g(n, f(n))$ for all $n \in \omega$.*

Proof. We first prove existence. Call a set $Z \subseteq \omega \times A$ *sufficient* if $(0, b) \in Z$ and for all $(n, a) \in Z$, we have $(S(n), g(n, a)) \in Z$. Notice that sufficient sets exists (since $\omega \times A$ is sufficient). Let

$$Y = \{(n, a) \in \omega \times A : (n, a) \in Z \text{ for every sufficient set } Z\}.$$

We first show that Y is sufficient. Notice that $(0, b) \in Y$ because $(0, b) \in Z$ for every sufficient set Z. Let $(n, a) \in Y$ be arbitrary. For any sufficient set Z, we then have $(n, a) \in Z$, so $(S(n), g(n, a)) \in Z$. Therefore, $(S(n), g(n, a)) \in Z$ for every sufficient set Z, so $(S(n), g(n, a)) \in Y$. It follows that Y is sufficient.

We next show that for all $n \in \omega$, there exists a unique $a \in A$ such that $(n, a) \in Y$. Let

$$X = \{n \in \omega : \text{There exists a unique } a \in A \text{ with } (n, a) \in Y\}.$$

Since Y is sufficient, we know that $(0, b) \in Y$. Let $d \in A$ be arbitrary with $d \neq b$. Since the set $(\omega \times A) \backslash \{(0, d)\}$ is sufficient (because $S(n) \neq 0$ for all $n \in \omega$), it follows that $(0, d) \notin Y$. Therefore, there exists a unique $a \in A$ such that $(0, a) \in Y$ (namely, $a = b$), so $0 \in X$.

Now let $n \in X$ be arbitrary, and let c be the unique element of A such that $(n, c) \in Y$. Since Y is sufficient, we have $(S(n), g(n, c)) \in Y$. Let $d \in A$ be arbitrary with $d \neq g(n, c)$. We then have that $Y \backslash \{(S(n), d)\}$ is sufficient (otherwise, there exists $a \in A$ such that $(n, a) \in Y$ and $g(n, a) = d$, contrary to

the fact that in this case we have $a = c$ by induction), so by definition of Y it follows that $Y \subseteq Y \backslash \{(S(n), d)\}$. Hence, $(S(n), d) \notin Y$. Therefore, there exists a unique $a \in A$ such that $(S(n), a) \in Y$ (namely, $a = g(n, c)$), so $S(n) \in X$.

By induction, we conclude that $X = \omega$, so for all $n \in \omega$, there exists a unique $a \in A$ such that $(n, a) \in Y$. Let $f = Y$ and notice that $f : \omega \to A$ because $X = \omega$. Since Y is sufficient, we have $(0, b) \in Y$, so $f(0) = b$. Let $n \in \omega$ be arbitrary. Since $(n, f(n)) \in Y$ and Y is sufficient, it follows that $(S(n), g(n, f(n))) \in Y$, so $f(S(n)) = g(n, f(n))$.

We now prove uniqueness. Suppose that $f_1, f_2 : \omega \to A$ are arbitrary function with the following properties:

1. $f_1(0) = b$.
2. $f_2(0) = b$.
3. $f_1(S(n)) = g(n, f_1(n))$ for all $n \in \omega$.
4. $f_2(S(n)) = g(n, f_2(n))$ for all $n \in \omega$.

Let $X = \{n \in \omega : f_1(n) = f_2(n)\}$. Notice that $0 \in X$ because $f_1(0) = b = f_2(0)$. Suppose that $n \in X$ so that $f_1(n) = f_2(n)$. We then have

$$\begin{aligned} f_1(S(n)) &= g(n, f_1(n)) \\ &= g(n, f_2(n)) \\ &= f_2(S(n)), \end{aligned}$$

hence $S(n) \in X$. It follows by induction that $X = \omega$, so $f_1(n) = f_2(n)$ for all $n \in \omega$. Therefore, $f_1 = f_2$. □

As an example of how to use this result (assuming we already know how to multiply - see below), consider how to define the factorial function. We want to justify the existence of a unique function $f : \omega \to \omega$ such that $f(0) = 1$ and $f(S(n)) = f(n) \cdot S(n)$ for all $n \in \omega$. We can make this work as follows. Let $A = \omega$, $b = 1$, and define $g : \omega \times \omega \to \omega$ by letting $g(n, a) = S(n) \cdot a$ (here we are thinking that the second argument of g will contain the "accumulated" value $f(n)$). The theorem now gives the existence and uniqueness of a function $f : \omega \to \omega$ such that $f(0) = 1$ and $f(S(n)) = S(n) \cdot f(n)$ for all $n \in \omega$.

However, this begs the question of how to define multiplication. Let's start by thinking about how to define addition. The basic idea is to define it recursively. For any $m \in \omega$, we let $m + 0 = m$. If $m \in \omega$, and we know how to find $m + n$ for some fixed $n \in \omega$, then we should define $m + S(n) = S(m + n)$. It looks like an appeal to the above theorem is in order, but how do we treat the m that is fixed in the recursion? We need a slightly stronger version of the above theorem which allows a parameter to come along for the ride.

Theorem 9.5.2 (Step Recursive Definitions with Parameters on ω)**.** *Let A and P be sets, let $h : P \to A$, and let $g : P \times \omega \times A \to A$. There exists a unique function $f : P \times \omega \to A$ such that $f(p, 0) = h(p)$ for all $p \in P$, and $f(p, S(n)) = g(p, n, f(p, n))$ for all $p \in P$ and all $n \in \omega$.*

Proof. One could reprove this from scratch following the above outline, but we give a simpler argument using Collection. For each $p \in P$, define $g_p : \omega \times A \to A$ by letting $g_p(n, a) = g(p, n, a)$ for all $(n, a) \in \omega \times A$. Using the above results without parameters, for each fixed $p \in P$, there exists a unique function $f_p : \omega \to A$ such that $f_p(0) = h(p)$ and $f_p(S(n)) = g_p(n, f_p(n))$ for all $n \in \omega$. By Collection and Separation, we may form the set $\{f_p : p \in \omega\}$. Let f be the union of this set. It is then straightforward to check that f is the unique function satisfying the necessary properties. □

Definition 9.5.3. *Let $h : \omega \to \omega$ be defined by $h(m) = m$ and let $g : \omega \times \omega \times \omega \to \omega$ be defined by $g(m, n, a) = S(a)$. We denote the unique f from the previous theorem by $+$. Notice that $+ : \omega \times \omega \to \omega$, that $m + 0 = m$ for all $m \in \omega$, and that $m + S(n) = S(m + n)$ for all $m, n \in \omega$.*

Now that we have the definition of $+$, we can prove all of the basic "axiomatic" facts about the natural numbers with $+$ by induction. Here's a simple example.

Proposition 9.5.4. *$0 + n = n$ for all $n \in \omega$.*

Proof. The proof is by induction on n. For $n = 0$, simply notice that $0 + 0 = 0$. Suppose that $n \in \omega$ and $0 + n = n$. We then have $0 + S(n) = S(0 + n) = S(n)$. The result follows by induction. □

A slightly more interesting example is a proof that $+$ is associative.

Proposition 9.5.5. *For all $k, m, n \in \omega$, we have $(k + m) + n = k + (m + n)$.*

Proof. We fix $k, m \in \omega$, and prove the result by induction on n. Notice that $(k+m)+0 = k+m = k+(m+0)$. Suppose that we know the result for n, so that $(k + m) + n = k + (m + n)$. We then have

$$\begin{aligned}(k + m) + S(n) &= S((k + m) + n) \\ &= S(k + (m + n)) && \text{(by induction)} \\ &= k + S(m + n) \\ &= k + (m + S(n)).\end{aligned}$$

The result follows by induction. □

Definition 9.5.6. *Let $h\colon \omega \to \omega$ be defined by $h(m) = 0$ and let $g\colon \omega \times \omega \times \omega \to \omega$ be defined by $g(m, a, n) = a + m$. We denote the unique f from the previous theorem by $\cdot$. Notice that $\cdot\colon \omega \times \omega \to \omega$, that $m \cdot 0 = 0$ for all $m \in \omega$, and that $m \cdot S(n) = m \cdot n + m$ for all $m, n \in \omega$.*

From now on, we will present our recursive definitions in the usual mathematical style. For example, we define iterates of a function as follows.

Definition 9.5.7. *Let B be a set, and let $h\colon B \to B$ be a function. We define, for each $n \in \omega$, a function h^n by letting $h^0 = id_B$ and letting $h^{S(n)} = h \circ h^n$ for all $n \in \omega$.*

For each fixed $h\colon B \to B$, this definition can be justified by appealing to the theorem with $A = B^B$, $b = id_B$, and $g\colon A \times \omega \to \omega$ given by $g(a, n) = h \circ a$. However, we will content ourselves with the above more informal style when the details are straightforward and uninteresting.

The above notions of recursive definitions can only handle types of recursion where the value of $f(S(n))$ depends just on the previous value $f(n)$ (and also n). Thus, it is unable to deal with recursive definitions such as that used in defining the Fibonacci sequence where the value of $f(n)$ depends on the two previous values of f whenever $n \geq 2$. We can justify these more general types of recursions by carrying along all previous values of f in the inductive construction. Thus, instead of having our iterating function $g\colon A \times \omega \to A$, where we think of the first argument of g as carrying the current value $f(n)$, we will have an iterating function $g\colon A^{<\omega} \to A$, where we think of the first argument of g as carrying the finite sequence consisting of all values $f(m)$ for $m < n$. Thus, given such a g, we are seeking the existence and uniqueness of a function $f\colon \omega \to A$ such that $f(n) = g(f \restriction n)$ for all $n \in \omega$. Notice that in this framework, we no longer need to put forward a $b \in A$ as a starting place for f because we will have $f(0) = g(\emptyset)$. Also, we do not need to include a number argument in the domain of g because the current n in the iteration can recovered as the domain of the single argument of g.

Theorem 9.5.8 (Recursive Definitions on ω)**.** *Let A be a set and let $g\colon A^{<\omega} \to A$. There exists a unique function $f\colon \omega \to A$ such that $f(n) = g(f \restriction n)$ for all $n \in \omega$.*

Proof. We first prove existence. Call a set $Z \subseteq \omega \times A$ *sufficient* if for all $n \in \omega$ and all $q \in A^n$ such that $(k, q(k)) \in Z$ for all $k < n$, we have $(n, g(q)) \in Z$. Notice that sufficient sets exists (since $\omega \times A$ is sufficient). Let

$$Y = \{(n, a) \in \omega \times A : (n, a) \in Z \text{ for every sufficient set } Z\}.$$

We first show that Y is sufficient. Let $n \in \omega$ and $q \in A^n$ be arbitrary such that $(k, q(k)) \in Y$ for all $k < n$. For any sufficient set Z, we have $(k, q(k)) \in Z$ for all $k < n$, so $(n, g(q)) \in Z$. Therefore, $(n, g(q)) \in Z$ for every sufficient set Z, so $(n, g(q)) \in Y$. It follows that Y is sufficient.

We next show that for all $n \in \omega$, there exists a unique $a \in A$ such that $(n, a) \in Y$. Let

$$X = \{n \in \omega : \text{There exists a unique } a \in A \text{ such that } (n, a) \in Y\}.$$

Let $n \in \omega$ be arbitrary such that $k \in X$ for all $k < n$. Let $q = Y \cap (n \times A)$ and notice that $q \in A^n$. Since $(k, q(k)) \in Y$ for all $k < n$ and Y is sufficient, it follows that $(n, g(q)) \in Y$. Let $b \in A$ be arbitrary with $b \neq g(q)$. We then have that $Y \backslash \{(n, b)\}$ is sufficient (otherwise, there exists $p \in A^n$ such that $(k, p(k)) \in Y$ for all $k < n$ and $g(p) = b$, but this implies that $p = q$ and hence $b = a$), so by definition of Y it follows that $Y \subseteq Y \backslash \{(n, b)\}$. Hence, $(n, b) \notin Y$. Therefore, there exists a unique $a \in A$ such that $(n, a) \in Y$, so $n \in X$.

By induction, we conclude that $X = \omega$, so for all $n \in \omega$, there exists a unique $a \in A$ such that $(n, a) \in Y$. Let $f = Y$ and notice that $f : \omega \to A$ because $X = \omega$. Let $n \in \omega$ be arbitrary. Let $q = Y \cap (n \times A)$ and notice that $q \in A^n$ and $q = f \restriction n$. Since $(k, q(k)) \in Y$ for all $k < n$ and Y is sufficient, it follows that $(n, g(q)) \in Y$, so $f(n) = g(q) = g(f \restriction n)$.

We now prove uniqueness. Suppose that $f_1, f_2 : \omega \to A$ are arbitrary functions with the following properties:

1. $f_1(n) = g(f_1 \restriction n)$ for all $n \in \omega$.
2. $f_2(n) = g(f_2 \restriction n)$ for all $n \in \omega$.

Let $X = \{n \in \omega : f_1(n) = f_2(n)\}$. We prove by induction that $X = \omega$. Let $n \in \omega$ be arbitrary such that $k \in X$ for all $k < n$. We then have that $f_1 \restriction n = f_2 \restriction n$, hence

$$\begin{aligned} f_1(n) &= g(f_1 \restriction n) \\ &= g(f_2 \restriction n) \\ &= f_2(n), \end{aligned}$$

hence $n \in X$. It follows by induction that $X = \omega$, so $f_1(n) = f_2(n)$ for all $n \in \omega$. Therefore, $f_1 = f_2$. □

As above, there is a similar version when we allow parameters. If $f : P \times \omega \to A$ and $p \in P$, we use the notation f_p to denote the function $f_p : \omega \to A$ given by $f_p(n) = f(p, n)$ for all $n \in \omega$.

Theorem 9.5.9 (Recursive Definitions with Parameters on ω)**.** *Let A and P be sets and let $g : P \times A^{<\omega} \to A$. There exists a unique function $f : P \times \omega \to A$ such that $f(p, n) = g(p, f_p \restriction n)$ for all $p \in P$ and $n \in \omega$.*

Proof. Similar to the proof of Theorem 9.5.2. □

9.6 Infinite Sets and Infinite Powers

Theorem 9.6.1 (Cantor-Schröder-Bernstein)**.** *Let A and B be sets. If $A \preceq B$ and $B \preceq A$, then $A \approx B$.*

Proof. We may assume that A and B are disjoint (otherwise, we can work with $A \times \{0\}$ and $B \times \{1\}$, and transfer the result back to A and B). Fix injections $f : A \to B$ and $g : B \to A$. We say that an element $a \in A$ is *B-originating* if there exists $b_0 \in B$ and $n \in \omega$ such that $b_0 \notin \text{range}(f)$ and $a = (g \circ f)^n(g(b_0))$. Similarly,

we say that an element $b \in B$ is B-*originating* if there exists $b_0 \in B$ and $n \in \omega$ such that $b_0 \notin \text{range}(f)$ and $b = (f \circ g)^n(b_0)$. Let

$$h = \{(a, b) \in A \times B : \text{Either } a \text{ is not } B\text{-originating and } f(a) = b \text{ or } a \text{ is } B\text{-originating and } g(b) = a\}.$$

Notice that h is a function (because f is a function and g is injective), $\text{domain}(h) \subseteq A$, and $\text{range}(h) \subseteq B$. We first show that $\text{domain}(h) = A$. Let $a \in A$ be arbitrary. If a is not B-originating, then $(a, f(a)) \in h$, hence $a \in \text{domain}(h)$. Suppose that a is B-originating, and fix $b_0 \in B$ and $n \in \omega$ with $a = (g \circ f)^n(g(b_0))$. If $n = 0$, then $a = g(b_0)$, so $(a, b_0) \in h$ and hence $a \in \text{domain}(h)$. Suppose that $n \neq 0$ and fix $m \in \omega$ with $n = S(m)$. We then have

$$\begin{aligned} a &= (g \circ f)^{S(m)}(g(b_0)) \\ &= (g \circ f)(((g \circ f)^m(g(b_0)))) \\ &= g(f((g \circ f)^m(g(b_0)))). \end{aligned}$$

Therefore, $(a, f((g \circ f)^m(g(b_0)))) \in h$, and hence $a \in \text{domain}(h)$. It follows that $\text{domain}(h) = A$.

We now know that $h \colon A \to B$, and we need only show that h is a bijection. Let $a_1, a_2 \in A$ be arbitrary with $h(a_1) = h(a_2)$. We first show that either both a_1 and a_2 are B-originating or both a_1 and a_2 are not B-originating. Without loss of generality, suppose that a_1 is B-originating and a_2 is not, so that $a_1 = g(h(a_1))$ and $h(a_2) = f(a_2)$. Since a_1 is B-originating, we may fix $b_0 \in B$ and $n \in \omega$ such that $b_0 \notin \text{range}(f)$ and $a_1 = (g \circ f)^n(g(b_0))$. Notice that

$$\begin{aligned} (g \circ f)^n(g(b_0)) &= a_1 \\ &= g(h(a_1)) \\ &= g(h(a_2)) \\ &= g(f(a_2)) \\ &= (g \circ f)(a_2). \end{aligned}$$

If $n = 0$, this implies that $g(b_0) = g(f(a_2))$, hence $f(a_2) = b_0$ (because g is injective), contrary to the fact that $b_0 \notin \text{range}(f)$. Suppose that $n \neq 0$ and fix $m \in \omega$ with $S(m) = n$. We then have

$$\begin{aligned} (g \circ f)((g \circ f)^m(g(b_0))) &= (g \circ f)^n(g(b_0)) \\ &= (g \circ f)(a_2), \end{aligned}$$

hence $(g \circ f)^m(g(b_0)) = a_2$ (because $g \circ f$ is injective), contrary to the fact that a_2 is not B-originating. Therefore, either a_1 and a_2 are both B-originating or both a_1 and a_2 are both not B-originating. If a_1 and a_2 are both not B-originating, this implies that $f(a_1) = f(a_2)$, hence $a_1 = a_2$ because f is injective. If a_1 and a_2 are both B-originating, we then have $a_1 = g(h(a_1)) = g(h(a_2)) = a_2$. It follows that h is injective.

We finally show that h is surjective. Fix $b \in B$. Suppose first that b is B-originating, and fix $b_0 \in B$ and $n \in \omega$ such that $b_0 \notin \text{range}(f)$ and $b = (f \circ g)^n(b_0)$. We then have $g(b) = g((f \circ g)^n(b_0)) = (g \circ f)^n(g(b_0))$, hence $g(b) \in A$ is B-originating. It follows that $h(g(b)) = b$, so $b \in \text{range}(h)$. Suppose now that b is not B-originating. We then must have $b \in \text{range}(f)$, so we may fix $a \in A$ with $f(a) = b$. If a is B-originating, we may fix $b_0 \in B$ and $n \in \omega$ such that $b_0 \notin \text{range}(f)$ and $a = (g \circ f)^n(g(b_0))$, and notice that $(f \circ g)^{S(n)}(b_0) = f((g \circ f)^n(g(b_0))) = f(a) = b$, contrary to the fact that b is not B-originating. Therefore, a is not B-originating, so $h(a) = f(a) = b$, and hence $b \in \text{range}(h)$. It follows that h is surjective. □

Definition 9.6.2. *Let A and B be sets. We write $A \prec B$ to mean that $A \preceq B$ and $A \not\approx B$.*

Definition 9.6.3. *Let A be a set.*

1. *A is* countably infinite *if $A \approx \omega$.*

2. *A is* countable *if A is either finite or countably infinite.*

3. *A is* uncountable *if A is not countable.*

Proposition 9.6.4. *Let A be a set. The following are equivalent:*

1. *A is countable.*

2. *$A \preceq \omega$.*

3. *There is a surjection $g\colon \omega \to A$.*

Proof. Exercise. □

Given a set A and an natural number $n \in \omega$, we defined A^n be the set of all functions from n to A. In fact, there is no reason to restrict to powers that are natural numbers. In general, we want to define A^B to be the set of all functions from B to A. We can certainly make this definition, but it is the first instance where we really need to use Power Set.

Proposition 9.6.5. *Let A and B be sets. There is a unique set, denoted by A^B, such that for all f, we have $f \in A^B$ if and only if $f\colon B \to A$.*

Proof. Notice that if $f\colon B \to A$, then $f \subseteq B \times A$, hence $f \in \mathcal{P}(B \times A)$. Therefore, $A^B = \{f \in \mathcal{P}(B \times A) : f$ is a function, $\text{domain}(f) = B$, and $\text{range}(f) = A\}$. As usual, uniqueness follows from Extensionality. □

Theorem 9.6.6. *For any set A, we have $A \prec \mathcal{P}(A)$.*

Proof. First, define a function $f\colon A \to \mathcal{P}(A)$ by letting $f(a) = \{a\}$ for every $a \in A$. Notice that f is an injection, hence $A \preceq \mathcal{P}(A)$. We next show that $A \not\approx \mathcal{P}(A)$ by showing that there is no bijection $f\colon A \to \mathcal{P}(A)$. Let $f\colon A \to \mathcal{P}(A)$ be an arbitrary function. Let $B = \{a \in A : a \notin f(a)\}$, and notice that $B \in \mathcal{P}(A)$. Suppose that $B \in \text{range}(f)$, and fix $b \in A$ with $f(b) = B$. We then have $b \in f(b) \leftrightarrow b \in B \leftrightarrow b \notin f(b)$, a contradiction. It follows that $B \notin \text{range}(f)$, hence f is not surjective. Therefore, $A \prec \mathcal{P}(A)$. □

9.7 Exercises

1. Let A and B be sets. Suppose that $a \in A$ and $b \in B$. Show that $(a, b) \in \mathcal{P}(\mathcal{P}(A \cup B))$ where $\mathcal{P}(x)$ is the power set of x. This allows one to construct $A \times B$ using Power Set (and Separation) instead of Collection.

2. Show that the class of all ordered pairs is a proper class.

3. (a) Let $m, n \in \omega$. Prove that if $S(m) = S(n)$, then $m = n$.
   (b) Show that for every $n \in \omega$ with $n \neq 0$, there exists a unique $m \in \omega$ with $S(m) = n$.
   (c) Prove that $m + n = n + m$ for all $m, n \in \omega$.

   Hint for c: It may help to first prove that $1 + n = S(n)$ for every $n \in \omega$.

4. Induction Variants:

   (a) Let $X \subseteq \omega$ and let $k \in \omega$. Suppose that $k \in X$ and whenever $n \in X$, we have $S(n) \in X$. Show that $n \in X$ for all $n \in \omega$ with $n \geq k$.

   (b) Let $X \subseteq \omega$ and let $k \in \omega$. Suppose that $0 \in X$ and whenever $n \in X$ and $n < k$, we have $S(n) \in X$. Show that for all $n \in \omega$, if $n \leq k$, then $n \in X$.

(c) Let $X \subseteq \omega \times \omega$. Suppose that for all $(m, n) \in \omega \times \omega$, if $(k, \ell) \in X$ for all $k, \ell \in \omega$ with either $k < m$ or $(k = m$ and $\ell < n)$, then $(m, n) \in X$. Prove that $X = \omega \times \omega$.

5. (a) Show that if A is a finite set and $b \notin A$, then $A \cup \{b\}$ is finite with $|A \cup \{b\}| = |A| + 1$.

   (b) Show that if A and B are finite and disjoint (i.e. $A \cap B = \emptyset$), then $A \cup B$ is finite with $|A \cup B| = |A| + |B|$.

6. (a) Explain how to define the function $f(n) = 2^n$ formally using Theorem 9.5.1.

   (b) Show that if A is finite with $|A| = n$, then $\mathcal{P}(A)$ is finite with $|\mathcal{P}(A)| = 2^n$.

Chapter 10

Ordinals, Cardinals, and Choice

10.1 Well-Orderings

The ability to do induction and make definitions by recursion on ω was essential to developing the basic properties of the natural numbers. With such success, we now want to push inductive arguments and recursive constructions to other structures. In Chapter 2, we successfully generalized the "step" versions of induction and recursive to other contexts where we generate elements one at a time. Now, we seek to generalize the ""order" versions. The key property underlying the order versions is the fact that $<$ is a well-ordering on ω. In fact, it is straightforward to see translate order induction to any well-ordering.

Proposition 10.1.1 (Induction on Well-Orderings)**.** *Let $(W, <)$ be a well-ordering.*

1. *Suppose that X is set and for all $z \in W$, if $y \in X$ for all $y < z$, then $z \in X$. We then have $W \subseteq X$.*
2. *For any formula $\varphi(\mathsf{z}, \vec{\mathsf{p}})$, we have the sentence*

$$\forall \vec{\mathsf{p}}((\forall \mathsf{z} \in W)((\forall \mathsf{y} < \mathsf{z})\varphi(\mathsf{y}, \vec{\mathsf{p}}) \to \varphi(\mathsf{z}, \vec{\mathsf{p}})) \to (\forall \mathsf{z} \in W)\varphi(\mathsf{z}, \vec{\mathsf{p}}))$$

3. *Suppose that $\boldsymbol{C}$ is a class and for all $z \in W$, if $y \in \boldsymbol{C}$ for all $y < z$, then $z \in \boldsymbol{C}$. We then have $W \subseteq \boldsymbol{C}$.*

Proof.

1. Suppose that $W \nsubseteq X$ so that $W \backslash X \neq \emptyset$. Since $(W, <)$ is a well-ordering, there exists $z \in W \backslash X$ such that for all $y \in W \backslash X$, either $z = y$ or $z < y$. Therefore, for all $y \in W$ with $y < z$, we have $y \in X$ (because $y \notin W \backslash X$). It follows from assumption that $z \in X$, contradicting the fact that $z \in W \backslash X$. Thus, it must be the case that $W \subseteq X$.

2. This follows from part 1 using Separation. Fix sets $\vec{q}$, and suppose that

$$(\forall z \in W)((\forall y < z)\varphi(y, \vec{q}) \to \varphi(z, \vec{q}))$$

   Let $X = \{z \in W : \varphi(n, \vec{q})\}$. Suppose that $z \in W$ and $y \in X$ for all $y < z$. We then have $(\forall y < z)\varphi(y, \vec{q})$, hence $\varphi(z, \vec{q})$ by assumption, so $z \in X$. It follows from part 1 that $W \subseteq X$. Therefore, we have $(\forall z \in W)\varphi(z, \vec{q})$.

3. This is just a restatement of 2 using the language of classes.

□

This is all well and good, but are there other interesting well-orderings other than ω (and every $n \in \omega$)? Well, any well-ordering has a smallest element. If there are any elements remaining, there must be a next smallest element. Again, if there are any elements remaining, there must be a next smallest element, and so on. In other words, any infinite well-ordering begins with a piece that looks like ω.

However, we can build another "longer" well-ordering by taking ω, and adding a new element which is greater than every element of ω. This can be visualized by thinking of the following subset of $\mathbb{R}$:

$$A = \left\{1 - \frac{1}{n} : n \in \omega \backslash \{0\}\right\} \cup \{1\}.$$

It's a simple exercise to check that A, under the natural ordering inherited from $\mathbb{R}$, is a well-ordering. We can then add another new element which is greater than every element, and another and another and so on, to get a well-ordering that is a copy of ω with another copy of ω on top of the first. We can add a new element greater than all of these, and continue. These well-orderings "beyond" ω differ from ω (and all $n \in \omega$) in that they have points that are neither initial points nor immediate successors of other points.

Definition 10.1.2. *Let $(W, <)$ be a well-ordering, and let $z \in W$.*

1. *If $z \leq y$ for all $y \in W$, we call z the* initial point *(such a z is easily seen to be unique).*
2. *If there exists $y \in W$ such that there is no $x \in W$ with $y < x < z$, we call z a* successor point*.*
3. *If z is neither an initial point nor a successor point, we call z a* limit point*.*

A little thought will suggest that all well-orderings should be built up by starting at an initial point, taking successors (perhaps infinitely often), and then jumping to a limit point above everything previously. After all, if we already have an initial part that looks like ω, and we haven't exhausted the well-ordering, then there must be a least element not accounted for, and this is the first limit point. If we still haven't exhausted it, there is another least element, which is a successor, and perhaps another successor, and so on. If this doesn't finish off the well-ordering, there is another least element not accounted for which will be the second limit point. For example, as a subset of the real line, the set

$$A = \left\{1 - \frac{1}{n} : n \in \omega \backslash \{0\}\right\} \cup \left\{2 - \frac{1}{n} : n \in \omega \backslash \{0\}\right\} \cup \{2\}.$$

is a well-ordering (under the inherited ordering from $\mathbb{R}$) with two limit points (namely 1 and 2).

This idea makes it seem plausible that we can take any two well-orderings and compare them by running through this procedure until one of them runs out of elements. That is, if $(W_1, <_1)$ and $(W_2, <_2)$ are well-orderings, then either they are isomorphic, or one is isomorphic to an initial segment of the other. We now develop the tools to prove this result. We first show that we can make recursive definitions along well-orderings. The proof is basically the same as the proof of the Induction Principle on ω because the only important fact that allowed that argument to work was the property of the order $<$ on ω (not the fact that every element of ω was either an initial point or a successor point).

Definition 10.1.3. *Let $(W, <)$ be a well-ordering, and let $z \in W$. We let $W(z) = \{y \in W : y < z\}$.*

Definition 10.1.4. *Let $(W, <)$ be a well-ordering. A set $I \subseteq W$ is called an* initial segment *of W if $I \neq W$ and whenever $x \in I$ and $y < x$, we have $y \in I$.*

Proposition 10.1.5. *Suppose that $(W, <)$ is a well-ordering and I is an initial segment of W. There exists $z \in W$ with $I = W(z)$.*

Proof. Since I is an initial segment of W, we have $I \subseteq W$ and $I \neq W$. Therefore, $W \backslash I \neq \emptyset$. Since $(W, <)$ is a well-ordering, there exists $z \in W \backslash I$ such that $z \leq y$ for all $y \in W \backslash I$. We claim that $I = W(z)$.

- Let $y \in W(z)$ be arbitrary. Since $y < z$, we then have $y \notin W \backslash I$, so $y \in I$. Therefore, $W(z) \subseteq I$.
- Let $y \in I$ be arbitrary with $y \notin W(z)$. We then have $y \not< z$, so $y \geq z$ because $<$ is a well-ordering (and hence a linear ordering). Therefore, $z \in I$ because I is an initial segment, contradicting the fact that $z \in W \backslash I$. It follows that $I \subseteq W(z)$.

Combining these, it follows from Extensionality that $I = W(z)$. □

Definition 10.1.6. *Let $(W, <)$ be a well-ordering and let A be a set. We let*

$$A^{<W} = \{f \in \mathcal{P}(W \times A) : f \text{ is a function and } f \colon W(z) \to A \text{ for some } z \in W\}.$$

Theorem 10.1.7 (Recursive Definitions on Well-Orderings)**.** *Let $(W, <)$ be a well-ordering, let A be a set, and let $g \colon A^{<W} \to A$. There exists a unique function $f \colon W \to A$ such that $f(z) = g(f \restriction W(z))$ for all $z \in W$.*

Proof. We first prove existence. Call a set $Z \subseteq W \times A$ *sufficient* if for all $z \in W$ and all $q \in A^{W(z)}$ such that $(y, q(y)) \in Z$ for all $y < z$, we have $(z, g(q)) \in Z$. Notice that sufficient sets exist (since $W \times A$ is sufficient). Let

$$Y = \{(z, a) \in W \times A : (z, a) \in Z \text{ for every sufficient set } Z\}.$$

We first show that Y is sufficient. Let $z \in W$ and $q \in A^{W(z)}$ be arbitrary such that $(y, q(y)) \in Y$ for all $y < z$. For any sufficient set Z, we have $(y, q(y)) \in Z$ for all $y < z$, so $(z, g(q)) \in Z$. Therefore, $(z, g(q)) \in Z$ for every sufficient set Z, so $(z, g(q)) \in Y$. It follows that Y is sufficient.

We next show that for all $z \in W$, there exists a unique $a \in A$ such that $(z, a) \in Y$. Let

$$X = \{z \in W : \text{There exists a unique } a \in A \text{ such that } (z, a) \in Y\}.$$

Let $z \in W$ be arbitrary such that $y \in X$ for all $y < z$. Let $q = Y \cap (W(z) \times A)$ and notice that $q \in A^{W(z)}$. Since $(y, q(y)) \in Y$ for all $y < z$ and Y is sufficient, it follows that $(z, g(q)) \in Y$. Fix $b \in A$ with $b \neq g(q)$. We then have that $Y \backslash \{(z, b)\}$ is sufficient (otherwise, there exists $p \in A^{W(z)}$ such that $(y, p(y)) \in Y$ for all $y < z$ and $g(p) = b$, but this implies that $p = q$ and hence $b = a$), so by definition of Y it follows that $Y \subseteq Y \backslash \{(z, b)\}$. Hence, $(z, b) \notin Y$. Therefore, there exists a unique $a \in A$ such that $(z, a) \in Y$, so $z \in X$.

By induction, we conclude that $X = W$, so for all $z \in W$, there exists a unique $a \in A$ such that $(z, a) \in Y$. Let $f = Y$ and notice that $f \colon W \to A$ because $X = W$. Let $z \in W$ be arbitrary. Define $q \in A^{W(z)}$ by letting $q = Y \cap (W(z) \times A)$ and notice that $q = f \restriction W(z)$. Since $(y, q(y)) \in Y$ for all $y < z$ and Y is sufficient, it follows that $(z, g(q)) \in Y$, so $f(z) = g(q) = g(f \restriction W(z))$.

We now prove uniqueness. Suppose that $f_1, f_2 \colon W \to A$ are arbitrary functions with the following properties:

1. $f_1(z) = g(f_1 \restriction W(z))$ for all $z \in W$.
2. $f_2(z) = g(f_2 \restriction W(z))$ for all $z \in W$.

Let $X = \{z \in W : f_1(z) = f_2(z)\}$. We prove by induction that $X = W$. Let $z \in W$ and suppose that $y \in X$ for all $y < z$. We then have that $f_1 \restriction W(z) = f_2 \restriction W(z)$, hence

$$\begin{aligned} f_1(z) &= g(f_1 \restriction W(z)) \\ &= g(f_2 \restriction W(z)) \\ &= f_2(z), \end{aligned}$$

hence $z \in X$. It follows by induction that $X = W$, so $f_1(z) = f_2(z)$ for all $z \in W$. Therefore, $f_1 = f_2$. □

Definition 10.1.8. *Let $(W_1, <_1)$ and $(W_2, <_2)$ be well-orderings.*

1. *A function* $f\colon W_1 \to W_2$ *is* order-preserving *if whenever* $x, y \in W_1$ *and* $x <_1 y$, *we have* $f(x) <_2 f(y)$.

2. *A function* $f\colon W_1 \to W_2$ *is an* isomorphism *if it is bijective and order-preserving.*

3. *If* W_1 *and* W_2 *are isomorphic, we write* $W_1 \cong W_2$.

Proposition 10.1.9. *Suppose that* $(W, <)$ *is a well-ordering and* $f\colon W \to W$ *is order-preserving. We then have* $f(z) \geq z$ *for all* $z \in W$.

Proof. We prove the result by induction on W. Suppose that $z \in W$ and $f(y) \geq y$ for all $y < z$. Suppose instead that $f(z) < z$, and let $x = f(z)$. Since f is order-preserving and $x < z$, it follows that $f(x) < f(z) = x$, contradicting the fact that $f(y) \geq y$ for all $y < z$. Therefore, $f(z) \geq z$. The result follows by induction. □

Proposition 10.1.10. *Let* $(W_1, <_1)$ *and* $(W_2, <_2)$ *be well-orderings. If* $f\colon W_1 \to W_2$ *is an isomorphism, then* $f^{-1}\colon W_2 \to W_1$ *is also an isomorphism.*

Proof. Exercise. □

Corollary 10.1.11.

1. *If* $(W, <)$ *is a well-ordering and* $z \in W$, *then* $W \ncong W(z)$.

2. *If* $(W, <)$ *is a well-ordering, then its only automorphism is the identity.*

3. *If* $(W_1, <_1)$ *and* $(W_2, <_2)$ *are well-orderings, and* $W_1 \cong W_2$, *then the isomorphism from* W_1 *to* W_2 *is unique.*

Proof.

1. Suppose that $W \cong W(z)$ for some $z \in W$ and let $f\colon W \to W(z)$ be a witnessing isomorphism. Then $f\colon W \to W$ is order-preserving and $f(z) < z$ (because $f(z) \in W(z)$), contrary to Proposition 10.1.9.

2. Let $f\colon W \to W$ be an arbitrary automorphism of W. Let $z \in W$ be arbitrary. By Proposition 10.1.9, we have $f(z) \geq z$. Since $f^{-1}\colon W \to W$ is also an automorphism of W, Proposition 10.1.9 implies that $f^{-1}(f(z)) \geq f(z)$, hence $z \geq f(z)$. Combining $f(z) \geq z$ and $z \geq f(z)$, we conclude that $z = f(z)$. Since $z \in W$ was arbitrary, we conclude that f is the identity function.

3. Suppose that $f\colon W_1 \to W_2$ and $g\colon W_1 \to W_2$ are both isomorphisms. We then have that $g^{-1}\colon W_2 \to W_1$ is an isomorphism, hence $g^{-1} \circ f\colon W_1 \to W_1$ is an automorphism. Hence, by part b, we may conclude that $g^{-1} \circ f$ is the identity on W_1. It follows that $f = g$.

□

Theorem 10.1.12. *Let* $(W_1, <_1)$ *and* $(W_2, <_2)$ *be well-orderings. Exactly one of the following holds:*

1. $W_1 \cong W_2$.

2. *There exists* $z \in W_2$ *such that* $W_1 \cong W_2(z)$.

3. *There exists* $z \in W_1$ *such that* $W_1(z) \cong W_2$.

In each of the above cases, the isomorphism and the z *(if appropriate) are unique.*

Proof. We first prove that one of the three options holds. Fix a set a such that $a \notin W_1 \cup W_2$ (such an a exists by Proposition 9.1.4). Our goal is to define a function $f\colon W_1 \to W_2 \cup \{a\}$ recursively. Define $g\colon (W_2 \cup \{a\})^{<W_1} \to W_2 \cup \{a\}$ as follows. Let $q \in (W_2 \cup \{a\})^{<W_1}$ be arbitrary, and fix $z \in W_1$ such that $q\colon W_1(z) \to W_2 \cup \{a\}$. If $a \in \text{range}(q)$ or $\text{range}(q) = W_2$, let $g(q) = a$. Otherwise $\text{range}(q)$ is a proper subset of W_2, and we let $g(q)$ be the $<_2$-least element of $W_2 \backslash \text{range}(q)$. By Theorem 10.1.7, there is a unique $f\colon W_1 \to W_2 \cup \{a\}$ such that $f(z) = g(f \restriction W_1(z))$ for all $z \in W_1$.

Suppose first that $a \notin \text{range}(f)$ so that $f\colon W_1 \to W_2$. We begin by showing that $\text{range}(f \restriction W_1(z))$ is an initial segment of W_2 for all $z \in W_1$ by induction. Let $z \in W_1$ be arbitrary such that $\text{range}(f \restriction W_1(y))$ is an initial segment of W_2 for all $y < z$. We have three cases:

- *Case 1:* Suppose that z is the initial point of W_1. We then $\text{range}(f \restriction W_1(z)) = \emptyset$ is certainly an initial segment of W_2.
- *Case 2:* Suppose that z is a successor point of W_1. Fix $y \in W_1$ such that there is no $x \in W_1$ with $y < x < z$. By induction, we know that $\text{range}(f \restriction W_1(y))$ is an initial segment of W_2. Since $f(y) = g(f \restriction W_1(y))$ is the $<_2$-least element of $W_2 \backslash \text{range}(f \restriction W_1(y))$, it follows that $\text{range}(f \restriction W_1(z)) = \text{range}(f \restriction W_1(y)) \cup \{f(y)\}$ is an initial segment of W_2.
- *Case 3:* Suppose finally that z is a limit point of W_1. It then follows that $\text{range}(f \restriction W_1(z)) = \bigcup_{y<z} \text{range}(f \restriction W_1(y))$. Since every element of the union is an initial segment of W_2, it follows that $\text{range}(f \restriction W_1(z))$ is an initial segment of W_2 (note that it can't equal W_2 because $f(z) \neq a$).

Therefore, $\text{range}(f \restriction W_1(z))$ is an initial segment of W_2 for all $z \in W_1$ by induction. It follows that for all $y, z \in W_1$ with $y < z$, we have $f(y) < f(z)$ (because $\text{range}(f \restriction W_1(z))$ is an initial segment of W_1 and $f(y) \in \text{range}(f \restriction W_1(z))$), so f is order-preserving. This implies that f is an injection, so if $\text{range}(f) = W_2$, we have $W_1 \cong W_2$. Otherwise, $\text{range}(f)$ is an initial segment of W_2, so by Proposition 10.1.5 there is a $z \in W_2$ such that $W_1 \cong W_2(z)$.

Suppose now that $a \in \text{range}(f)$. Let $z \in W_1$ be the $<_1$-least element of W_1 such that $f(z) = a$. It then follows that $f \restriction W_1(z) : W_1(z) \to W_2$ is order-preserving by induction as above. Also, we must have $\text{range}(f \restriction W_1(z)) = W_2$ because $f(z) = a$. Therefore, $f \restriction W_1(z) : W_1(z) \to W_2$ is an isomorphism. This completes the proof that one of the above 3 cases must hold.

The uniqueness of the case, the isomorphism, and the z (if appropriate), all follow from Corollary 10.1.11 □

With this result in hand, we now know that any well-ordering is uniquely determined by its "length". The next goal is to find a nice system of representatives for the isomorphism classes of well-orderings. For that, we need to generalize the ideas that went into the construction of the natural numbers.

10.2 Ordinals

Our definition of the natural numbers had the advantage that the ordering was given by the membership relation $\in$. This feature allowed us to define successors easily and to think of a natural number n as the set of all natural numbers less than n. We now seek to continue this progression to measure well-orderings longer than ω. The idea is to define successors as in the case of the natural numbers, but now to take unions to achieve limit points.

The key property of ω (and each $n \in \omega$) that we want to use in our definition of ordinals is the fact that $\in$ well-orders ω (and each $n \in \omega$). We need one more condition to ensure that there are no "holes" or "gaps" in the set. For example, $\in$ well-orders the set $\{0, 2, 3, 5\}$, but we don't want to consider it as an ordinal because it skipped over 1 and 4. We therefore make the following definition.

Definition 10.2.1. *A set z is transitive if whenever x and y are sets such that $x \in y$ and $y \in z$, we have $x \in z$.*

Definition 10.2.2. *Let z be a set. We define a relation $\in_z$ on z by setting $\in_z = \{(x,y) \in z \times z : x \in y\}$.*

Definition 10.2.3. *An* ordinal *is a set α which is transitive and well-ordered by $\in_\alpha$.*

Our hard work developing the natural numbers gives us one interesting example of an ordinal.

Proposition 10.2.4. *ω is an ordinal.*

Proof. Proposition 9.2.11 says that ω is transitive, and Theorem 9.2.18 says that ω is well-ordered by $< = \in_\omega$. □

Proposition 10.2.5. *If α is an ordinal and $\beta \in \alpha$, then β is an ordinal.*

Proof. We first show that β is transitive. Let x and y be sets with $x \in y$ and $y \in \beta$. Since $y \in \beta$, $\beta \in \alpha$, and α is transitive, it follows that $y \in \alpha$. Since $x \in y$ and $y \in \alpha$, it follows that $x \in \alpha$. Now since $x, y, \beta \in \alpha$, $x \in y$, $y \in \beta$, and $\in_\alpha$ is transitive on α, we may conclude that $x \in \beta$. Therefore, β is transitive.

Notice that $\beta \subseteq \alpha$ because $\beta \in \alpha$ and α is transitive. Therefore, $\in_\beta$ is the restriction of $\in_\alpha$ to the subset $\beta \subseteq \alpha$. Since $\in_\alpha$ is a well-ordering on α, it follows that $\in_\beta$ is a well-ordering on β. Hence, β is an ordinal. □

Corollary 10.2.6. *Every $n \in \omega$ is an ordinal.*

Lemma 10.2.7. *If α is an ordinal, then $\alpha \notin \alpha$.*

Proof. Suppose that α is an ordinal and $\alpha \in \alpha$. Since $\alpha \in \alpha$, it follows that $\in_\alpha$ is not asymmetric on α, contradicting the fact that $\in_\alpha$ is a well-ordering on α. □

Proposition 10.2.8. *If α is an ordinal, then $S(\alpha)$ is an ordinal.*

Proof. We first show that $S(\alpha)$ is transitive. Suppose that $x \in y \in S(\alpha)$. Since $y \in S(\alpha) = \alpha \cup \{\alpha\}$, either $y \in \alpha$ or $y = \alpha$. Suppose first that $y \in \alpha$. We then have $x \in y \in \alpha$, so $x \in \alpha$ because α is transitive. Hence, $x \in S(\alpha)$. Suppose now that $y = \alpha$. We then have $x \in \alpha$ because $x \in y$, so $x \in S(\alpha)$.

We next show that $\in_{S(\alpha)}$ is transitive on $S(\alpha)$. Let $x, y, z \in S(\alpha)$ with $x \in y \in z$. Since $z \in S(\alpha)$, either $z \in \alpha$ or $z = \alpha$. Suppose first that $z \in \alpha$. We then have $y \in \alpha$ (since $y \in z \in \alpha$ and α is transitive), and hence $x \in \alpha$ (since $x \in y \in \alpha$ and α is transitive). Thus, $x, y, z \in \alpha$, so we may conclude that $x \in z$ using the fact that $\in_\alpha$ is transitive on α. Suppose now that $z = \alpha$. We then have $x \in \alpha = z$ because $x \in y \in \alpha$ and α is transitive.

We next show that $\in_{S(\alpha)}$ is asymmetric on $S(\alpha)$. Let $x \in S(\alpha)$. If $x \in \alpha$, then $x \notin x$ because $\in_\alpha$ is asymmetric on α. If $x = \alpha$, then $x \notin x$ by Lemma 10.2.7.

We now show that $\in_{S(\alpha)}$ is connected on $S(\alpha)$. Let $x, y \in S(\alpha)$. If $x \in \alpha$ and $y \in \alpha$, then either $x \in y$, $x = y$, or $y \in x$ because $\in_\alpha$ is connected on α. If $x = \alpha$ and $y = \alpha$, we clearly have $x = y$. Otherwise, one of x, y equals α, and the other is an element of α, if which case we're done.

Finally, suppose that $X \subseteq S(\alpha)$ and $X \neq \emptyset$. If $X \cap \alpha = \emptyset$, then we must have $X = \{\alpha\}$, in which case X clearly has a $\in_{S(\alpha)}$-least element. Suppose that $X \cap \alpha \neq \emptyset$. Since $X \cap \alpha \subseteq \alpha$ is nonempty and $\in_\alpha$ is a well-ordering on α, there exists a $\in_\alpha$-least element β in $X \cap \alpha$. For any $\gamma \in X$, either $\gamma \in \alpha$ in which case we have have either $\beta = \gamma$ or $\beta \in \gamma$ by choice of β, or $\gamma = \alpha$ in which case $\beta \in \gamma$ (because $\beta \in \alpha$). Therefore, X has a $\in_{S(\alpha)}$-least element. □

Proposition 10.2.9. *Suppose that α and β are ordinals. We then have $\alpha \subseteq \beta$ if and only if either $\alpha = \beta$ or $\alpha \in \beta$.*

Proof. ($\Leftarrow$) If $\alpha = \beta$, then clearly $\alpha \subseteq \beta$ and if $\alpha \in \beta$ we can use the fact that β is transitive to conclude that $\alpha \subseteq \beta$.

($\Rightarrow$) Suppose that $\alpha \subseteq \beta$ and $\alpha \neq \beta$. Notice that $\beta \backslash \alpha$ is an nonempty subset of β, so there exists a $\in_\beta$-least element of $\beta \backslash \alpha$, call it z. We show that $\alpha = z$, hence $\alpha \in \beta$. We first show that $z \subseteq \alpha$. Let $x \in z$.

Since $z \in \beta$ and β is transitive, we have $x \in \beta$. Since $x \in z$, we can not have $x \in \beta \backslash \alpha$ by choice of z, so $x \in \alpha$. Thus, $z \subseteq \alpha$. We next show that $\alpha \subseteq z$. Let $x \in \alpha$. Since $\alpha \subseteq \beta$, we have $x \in \beta$. Using the fact that $x, z \in \beta$ and $\in_\beta$ is connected on β, we know that either $x \in z$, $x = z$, or $z \in x$. We can not have $x = z$ because $x \in \alpha$ and $z \in \beta \backslash \alpha$. Also, we can not have $z \in x$, because if $z \in x$ we can also conclude that $z \in \alpha$ (because $z \in x \in \alpha$ and α is transitive), contradicting the fact that $z \in \beta \backslash \alpha$. Thus, $\alpha \subseteq z$. It follows that $z = \alpha$ (by Extensionality), so $\alpha \in \beta$. □

Proposition 10.2.10. *Suppose that α and β are ordinals. Exactly one of $\alpha \in \beta$, $\alpha = \beta$, or $\beta \in \alpha$ holds.*

Proof. We first show that at least one of $\alpha \in \beta$, $\alpha = \beta$, $\beta \in \alpha$ holds. We first claim that $\alpha \cap \beta$ is an ordinal. If $x \in y \in \alpha \cap \beta$, then $x \in y \in \alpha$ and $x \in y \in \beta$, so $x \in \alpha$ and $x \in \beta$ (because α and β are transitive), and hence $x \in \alpha \cap \beta$. Thus, $\alpha \cap \beta$ is transitive. Notice that $\in_{\alpha \cap \beta}$ is the restriction of $\in_\alpha$ to the subset $\alpha \cap \beta \subseteq \alpha$. Since $\in_\alpha$ is a well-ordering on α, it follows that $\in_{\alpha \cap \beta}$ is a well-ordering on $\alpha \cap \beta$. Hence, $\alpha \cap \beta$ is an ordinal.

Now we have $\alpha \cap \beta \subseteq \alpha$ and $\alpha \cap \beta \subseteq \beta$. If $\alpha \cap \beta \neq \alpha$ and $\alpha \cap \beta \neq \beta$, then $\alpha \cap \beta \in \alpha$ and $\alpha \cap \beta \in \beta$ by Proposition 10.2.9, hence $\alpha \cap \beta \in \alpha \cap \beta$, contrary to Lemma 10.2.7. Therefore, either $\alpha \cap \beta = \alpha$ or $\alpha \cap \beta = \beta$. If $\alpha \cap \beta = \alpha$, we then have $\alpha \subseteq \beta$, hence either $\alpha = \beta$ or $\alpha \in \beta$ by Proposition 10.2.9. Similarly, if $\alpha \cap \beta = \beta$, we then have $\beta \subseteq \alpha$, hence either $\beta = \alpha$ or $\beta \in \alpha$ by Proposition 10.2.9. Thus, in any case, at least one $\alpha \in \beta$, $\alpha = \beta$, or $\beta \in \alpha$ holds.

We finish by showing that exactly one of $\alpha \in \beta$, $\alpha = \beta$, or $\beta \in \alpha$ holds. If $\alpha \in \beta$ and $\alpha = \beta$, then $\alpha \in \alpha$, contrary to Lemma 10.2.7. Similarly, if $\alpha = \beta$ and $\beta \in \alpha$, then $\beta \in \beta$, contrary to Lemma 10.2.7. Finally, if $\alpha \in \beta$ and $\beta \in \alpha$, then $\alpha \in \alpha$ (because α is transitive), contrary to Lemma 10.2.7. □

Definition 10.2.11. *If α and β are ordinals, we write $\alpha < \beta$ to mean that $\alpha \in \beta$.*

Proposition 10.2.12. *Let α and β be arbitrary ordinals. We have $\alpha < S(\beta)$ if and only if $\alpha \leq \beta$.*

Proof. Notice that $S(\beta)$ is an ordinal by Proposition 10.2.8. Now

$$\begin{aligned} \alpha < S(\beta) &\Leftrightarrow \alpha \in S(\beta) \\ &\Leftrightarrow \alpha \in \beta \cup \{\beta\} \\ &\Leftrightarrow \text{Either } \alpha \in \beta \text{ or } \alpha \in \{\beta\} \\ &\Leftrightarrow \text{Either } \alpha < \beta \text{ or } \alpha = \beta \\ &\Leftrightarrow \alpha \leq \beta. \end{aligned}$$

□

Proposition 10.2.13. *Suppose that α and β are ordinals. If $\alpha \cong \beta$ as well-orderings, then $\alpha = \beta$.*

Proof. If $\alpha \neq \beta$, then either $\alpha < \beta$ or $\beta < \alpha$ by Proposition 10.2.10. Suppose without loss of generality that $\beta < \alpha$. We then have that the well-ordering β is an initial segments of the well-ordering α (in the notation for well-orderings, we have $\beta = \alpha(\beta)$), hence $\alpha \ncong \beta$ by Corollary 10.1.11. □

By the above results, it seems that we are in a position to say that $<$ is a linear ordering on the collection of all ordinals. However, there is a small problem here. We do not know that the class of all ordinals is a set. In fact, we will see below that the collection of all ordinals is a proper class.

Definition 10.2.14. ***ORD*** *is the class of all ordinals.*

We first establish that nonempty sets of ordinals have least elements.

Proposition 10.2.15. *If A is a nonempty subset of **ORD**, then A has a least element. Furthermore the least element is given by $\bigcap A$.*

Proof. Since $A \neq \emptyset$, we may fix an ordinal $\alpha \in A$. If $A \cap \alpha = \emptyset$, then for any $\beta \in A$, we can not have $\beta \in \alpha$, hence either $\alpha = \beta$ or $\alpha \in \beta$ by Proposition 10.2.10. Suppose that $A \cap \alpha \neq \emptyset$. Since $A \cap \alpha \subseteq \alpha$ is nonempty, it has an $\in_\alpha$-least element, call it δ. Let $\beta \in A$ and notice that β is an ordinal. By Proposition 10.2.10, either $\beta \in \alpha$, $\beta = \alpha$, or $\alpha \in \beta$. If $\beta \in \alpha$, then $\beta \in A \cap \alpha$, so either $\delta = \beta$ or $\delta \in \beta$ by choice of δ. If $\beta = \alpha$, then $\delta \in \beta$ because $\delta \in \alpha$. If $\alpha \in \beta$, we then have $\delta \in \alpha \in \beta$, so $\delta \in \beta$ because β is transitive. It follows that δ is the least element of A.

Therefore, we know that A has a least element, call it δ. Since $\delta \in A$, we certainly have $\bigcap A \subseteq \delta$. For all $\alpha \in A$, we then have either $\delta = \alpha$ or $\delta \in \alpha$, hence $\delta \subseteq \alpha$ by Proposition 10.2.9. Therefore, $\delta \subseteq \bigcap A$. It follows that $\delta = \bigcap A$. □

Proposition 10.2.16. *If A is a subset of **ORD**, then $\bigcup A$ is an ordinal. Furthermore, we have $\bigcup A = \sup A$, i.e. $\alpha \leq \bigcup A$ for all $\alpha \in A$ and $\bigcup A \leq \beta$ whenever β is an ordinal with $\beta \geq \alpha$ for all $\alpha \in A$.*

Proof. We first show that $\bigcup A$ is transitive. Suppose that $x \in y \in \bigcup A$. Since $y \in \bigcup A$, there exists $\alpha \in A$, necessarily an ordinal, such that $y \in \alpha \in A$. Since α is transitive and $x \in y \in \alpha$, we can conclude that $x \in \alpha$. It follows that $x \in \bigcup A$. Hence, $\bigcup A$ is transitive.

We next show that $\in_{\bigcup A}$ is transitive on $\bigcup A$. Let $x, y, z \in \bigcup A$ with $x \in y \in z$. Since $z \in \bigcup A$, there exists $\alpha \in A$, necessarily an ordinal, such that $z \in \alpha \in A$. Since $z \in \alpha$ and α is an ordinal, we may use Proposition 10.2.5 to conclude that z is an ordinal. Thus, z is transitive, so we may use the fact that $x \in y \in z$ to conclude that $x \in z$.

We next show that $\in_{\bigcup A}$ is asymmetric on $\bigcup A$. Let $x \in \bigcup A$ and fix $\alpha \in A$, necessarily an ordinal, such that $x \in \alpha \in A$. Using Proposition 10.2.5 again, it follows that x is an ordinal, hence $x \notin x$ by Lemma 10.2.7.

We now show that $\in_{\bigcup A}$ is connected on $\bigcup A$. Let $x, y \in \bigcup A$. Fix $\alpha, \beta \in A$, necessarily ordinals, such that $x \in \alpha \in A$ and $y \in \beta \in A$. Again, using Proposition 10.2.5, we may conclude that x and y are ordinals, hence either $x \in y$, $x = y$, or $y \in x$ by Proposition 10.2.10

Finally, suppose that $X \subseteq \bigcup A$ and $X \neq \emptyset$. Notice that for any $y \in X$, there exists $\alpha \in A$, necessarily an ordinal, such that $y \in \alpha \in A$, and hence y is an ordinal by Proposition 10.2.10. Therefore, X is a nonempty subset of **ORD**, so by Proposition 10.2.15 we may conclude that X has a least element (with respect to $\in_{\bigcup A}$).

We now show that $\bigcup A = \sup A$. Suppose that $\alpha \in A$. For any $\beta \in \alpha$, we have $\beta \in \alpha \in A$, hence $\beta \in \bigcup A$. It follows that $\alpha \subseteq \bigcup A$, hence $\alpha \leq \bigcup A$ by Proposition 10.2.9. Thus, $\bigcup A$ is an upper bound for A. Suppose that γ is an upper bound for A, i.e. γ is an ordinal and $\alpha \leq \gamma$ for all $\alpha \in A$. For any $\beta \in \bigcup A$, we may fix $\alpha \in A$ such that $\beta \in \alpha$ and notice that $\beta \in \alpha \subseteq \gamma$, so $\beta \in \gamma$. It follows that $\bigcup A \subseteq \gamma$, hence $\bigcup A \leq \gamma$ by Proposition 10.2.9. Therefore, $\bigcup A = \sup A$. □

Proposition 10.2.17. ***ORD** is a proper class.*

Proof. Suppose that **ORD** is a set, so that there is a set O such that α is an ordinal if and only $\alpha \in O$. In this case, O is a transitive set (by Proposition 10.2.5) which is well-ordered by $\in_O$ (transitivity follows from the fact that ordinals are transitive sets, asymmetry follows from Lemma 10.2.7, connectedness follows from Proposition 10.2.10, and the fact that every nonempty subset has a least element is given by Proposition 10.2.15). Therefore, O is an ordinal and so it follows that $O \in O$, contrary to Lemma 10.2.7. Hence, **ORD** is not a set. □

Since **ORD** is a proper class, there are subclasses of **ORD** which are not subsets of **ORD**. We therefore extend Proposition 10.2.15 to the case of nonempty subclasses of **ORD**. The idea is that if we fix an $\alpha \in \mathbf{C}$, then $\alpha \cap \mathbf{C}$ becomes a set of ordinals, so we can apply the above result.

Proposition 10.2.18. *If **C** is a nonempty subclass of **ORD**, then **C** has a least element.*

Proof. Since $\mathbf{C} \neq \emptyset$, we may fix an ordinal $\alpha \in \mathbf{C}$. If $\mathbf{C} \cap \alpha = \emptyset$, then for any $\beta \in \mathbf{C}$, we can not have $\beta \in \alpha$, hence either $\alpha = \beta$ or $\alpha \in \beta$ by Proposition 10.2.10. Suppose that $\mathbf{C} \cap \alpha \neq \emptyset$. In this case, $\mathbf{C} \cap \alpha$ is a nonempty set of ordinals by Separation, hence $\mathbf{C} \cap \alpha$ has a least element δ by Proposition 10.2.15. It now follows easily that δ is the least element of $\mathbf{C}$. □

Proposition 10.2.19 (Induction on **ORD**)**.** *Suppose that $\boldsymbol{C} \subseteq \boldsymbol{ORD}$ and that for all ordinals α, if $\beta \in \boldsymbol{C}$ for all $\beta < \alpha$, then $\alpha \in \boldsymbol{C}$. We then have $\boldsymbol{C} = \boldsymbol{ORD}$.*

Proof. Suppose that $\mathbf{C} \subsetneq \mathbf{ORD}$. Let $\mathbf{B} = \mathbf{ORD} \setminus \mathbf{C}$ and notice that $\mathbf{B}$ is a nonempty class of ordinals. By Proposition 10.2.18, it follows that $\mathbf{B}$ has a least element, call it α. For all $\beta < \alpha$, we then have $\beta \notin \mathbf{B}$, hence $\beta \in \mathbf{C}$. By assumption, this implies that $\alpha \in \mathbf{C}$, a contradiction. It follows that $\mathbf{C} = \mathbf{ORD}$. □

This gives a way to do "strong induction" on the ordinals, but there is a slightly more basic version.

Definition 10.2.20. *Let α be an ordinal.*

- *We say that α is a* successor ordinal *if there exists an ordinal β with $\alpha = S(\beta)$.*
- *We say that α is a* limit ordinal *if $\alpha \neq 0$ and α is not a successor ordinal.*

Notice that α is a limit ordinal if and only if $\alpha \neq 0$, and whenever $\beta < \alpha$, we have $S(\beta) < \alpha$. For example, ω is a limit ordinal. In an inductive argument, we can't get around looking at many previous values at limit ordinals, but we can by with just looking at the previous ordinal in the case of successors.

Proposition 10.2.21 (Step/Limit Induction on **ORD**)**.** *Suppose that $\boldsymbol{C} \subseteq \boldsymbol{ORD}$ with the following properties:*

1. *$0 \in \boldsymbol{C}$.*
2. *Whenever $\alpha \in \boldsymbol{C}$, we have $S(\alpha) \in \boldsymbol{C}$.*
3. *Whenever α is a limit ordinal and $\beta \in \boldsymbol{C}$ for all $\beta < \alpha$, we have $\alpha \in \boldsymbol{C}$.*

We then have $\boldsymbol{C} = \boldsymbol{ORD}$.

Proof. Suppose that $\mathbf{C} \subsetneq \mathbf{ORD}$. Let $\mathbf{B} = \mathbf{ORD} \setminus \mathbf{C}$ and notice that $\mathbf{B}$ is a nonempty class of ordinals. By Proposition 10.2.18, it follows that $\mathbf{B}$ has a least element, call it α. We can't have $\alpha = 0$ because $0 \in \mathbf{C}$. Also, it is not possible that α is a successor, say $\alpha = S(\beta)$, because if so, then $\beta \notin B$ (because $\beta < \alpha$), so $\beta \in \mathbf{C}$, hence $\alpha = S(\beta) \in \mathbf{C}$. Finally, suppose that α is a limit. Then for for all $\beta < \alpha$, we have $\beta \notin \mathbf{B}$, hence $\beta \in \mathbf{C}$. By assumption, this implies that $\alpha \in \mathbf{C}$, a contradiction. It follows that $\mathbf{C} = \mathbf{ORD}$. □

Theorem 10.2.22 (Recursive Definitions on **ORD**)**.** *Let $\boldsymbol{G} : \boldsymbol{V} \to \boldsymbol{V}$ be a class function. There exists a unique class function $\boldsymbol{F} : \boldsymbol{ORD} \to \boldsymbol{V}$ such that $\boldsymbol{F}(\alpha) = \boldsymbol{G}(\boldsymbol{F} \restriction \alpha)$ for all $\alpha \in \boldsymbol{ORD}$.*

Theorem 10.2.23 (Recursive Definitions with Parameters on **ORD**)**.** *Let $\boldsymbol{P}$ be a class and let $\boldsymbol{G} : \boldsymbol{P} \times \boldsymbol{V} \to \boldsymbol{V}$ be a class function. There exists a unique class function $\boldsymbol{F} : \boldsymbol{P} \times \boldsymbol{ORD} \to \boldsymbol{V}$ such that $\boldsymbol{F}(p, \alpha) = \boldsymbol{G}(\boldsymbol{F}_p \restriction \alpha)$ for all $p \in \boldsymbol{P}$ and all $\alpha \in \boldsymbol{ORD}$.*

Theorem 10.2.24. *Let $(W, <)$ be a well-ordering. There exists a unique ordinal α such that $W \cong \alpha$.*

Proof. Fix a set a such that $a \notin W$ (such an a exists by Proposition 9.1.4). We define a class function $\mathbf{F} : \mathbf{ORD} \to W \cup \{a\}$ recursively as follows. If $a \in \text{range}(\mathbf{F} \restriction \alpha)$ or $\text{range}(\mathbf{F} \restriction \alpha) = W$, let $\mathbf{F}(\alpha) = a$. Otherwise, $\text{range}(\mathbf{F} \restriction \alpha) \subsetneq W$, and we let $\mathbf{F}(\alpha)$ be the least element of $W \setminus \text{range}(\mathbf{F} \restriction \alpha)$.

Since **ORD** is a proper class, it follows from Proposition 9.3.5 that $\mathbf{F}$ is not injective. From this it follows that $a \in \text{range}(\mathbf{F})$ (otherwise, a simple inductive proof gives that $\mathbf{F}$ would have to be injective). Let α be the least ordinal such that $\mathbf{F}(\alpha) = a$. Now it is straightforward to prove (along the lines of the proof of Theorem 10.1.12) that $\mathbf{F} \restriction \alpha : \alpha \to W$ is an isomorphism.

Uniqueness follows from Proposition 10.2.13 □

Definition 10.2.25. *Let $(W, <)$ be a well-ordering. The unique ordinal α such that $W \cong \alpha$ is called the* order-type *of $(W, <)$.*

10.3 Arithmetic on Ordinals

Now that we have the ability to define functions recursively on all of the ordinals, we can extend our definitions of addition, multiplication, and exponentiation of natural numbers into the transfinite.

Definition 10.3.1. *We define ordinal addition (that is a class function $+$: $\boldsymbol{ORD} \times \boldsymbol{ORD} \to \boldsymbol{ORD}$) recursively as follows.*

1. $\alpha + 0 = \alpha$.
2. $\alpha + S(\beta) = S(\alpha + \beta)$.
3. $\alpha + \beta = \bigcup\{\alpha + \gamma : \gamma < \beta\}$ *if β is a limit ordinal.*

Similarly, we define ordinal multiplication recursively as follows.

1. $\alpha \cdot 0 = 0$.
2. $\alpha \cdot S(\beta) = \alpha \cdot \beta + \alpha$.
3. $\alpha \cdot \beta = \bigcup\{\alpha \cdot \gamma : \gamma < \beta\}$ *if β is a limit ordinal.*

Finally, we define ordinal exponentiation recursively as follows.

1. $\alpha^0 = 1$.
2. $\alpha^{S(\beta)} = \alpha^\beta \cdot \alpha$.
3. $\alpha^\beta = \bigcup\{\alpha^\gamma : \gamma < \beta\}$ *if β is a limit ordinal.*

Notice that we have

$$\begin{aligned} \omega + 1 &= \omega + S(0) \\ &= S(\omega + 0) \\ &= S(\omega). \end{aligned}$$

On the other hand, since ω is a limit ordinal and $+$ is commutative on ω (by the homework), we have

$$\begin{aligned} 1 + \omega &= \bigcup\{1 + n : n < \omega\} \\ &= \bigcup\{n + 1 : n < \omega\} \\ &= \bigcup\{n + S(0) : n < \omega\} \\ &= \bigcup\{S(n + 0) : n < \omega\} \\ &= \bigcup\{S(n) : n \in \omega\} \\ &= \omega. \end{aligned}$$

Therefore, we have $\omega + 1 \neq 1 + \omega$, and hence addition of ordinals is *not* commutative in general. Moreover, notice that even though we have $0 < 1$, we do not have $0 + \omega < 1 + \omega$ (because $0 + \omega = \omega$ as well. In other words, addition on the right does not preserve the strict ordering relation. In contrast, addition on the left does preserve the ordering, as we now show. We start with the non-strict version.

Proposition 10.3.2. *Let α, β, and γ be ordinals. If $\beta \leq \gamma$, then $\alpha + \beta \leq \alpha + \gamma$.*

Proof. Fix arbitrary ordinals α and β. We prove by induction on γ that if $\beta \leq \gamma$, then $\alpha + \beta \leq \alpha + \gamma$. For the base case, notice that the statement is trivial when $\gamma = \beta$. For the successor step, let $\gamma \geq \beta$ be arbitrary such that $\alpha + \beta \leq \alpha + \gamma$. We then have

$$\begin{aligned} \alpha + \beta &\leq \alpha + \gamma \\ &< S(\alpha + \gamma) \\ &= \alpha + S(\gamma), \end{aligned}$$

so the statement is true for $S(\gamma)$. For the limit case, suppose that $\gamma > \beta$ is a limit ordinal with the property that $\alpha + \beta \leq \alpha + \delta$ whenever $\beta \leq \delta < \gamma$. We then have

$$\begin{aligned} \alpha + \beta &\leq \bigcup\{\alpha + \delta : \delta < \gamma\} && \text{(since } \beta < \gamma\text{)} \\ &= \alpha + \gamma, \end{aligned}$$

so the statement is true for γ. The result follows by induction. □

Proposition 10.3.3. *Let α, β, and γ be ordinals. We have $\beta < \gamma$ if and only if $\alpha + \beta < \alpha + \gamma$.*

Proof. Let α and β be arbitrary ordinals. Notice first that

$$\alpha + \beta < S(\alpha + \beta) = \alpha + S(\beta).$$

Now if γ is an arbitrary ordinal with $\gamma > \beta$, then we have $S(\beta) \leq \gamma$, hence

$$\alpha + \beta < \alpha + S(\beta) \leq \alpha + \gamma$$

by Proposition 10.3.2. Therefore, we have $\alpha + \beta < \alpha + \gamma$. For the converse, notice that if $\gamma < \beta$, then $\alpha + \gamma < \alpha + \beta$ by what we just proved. □

Proposition 10.3.4. *Let α and β be ordinals. If β is a limit ordinal, then $\alpha + \beta$ is a limit ordinal.*

Proof. Since β is a limit ordinal, we have

$$\alpha + \beta = \bigcup\{\alpha + \gamma : \gamma < \beta\}.$$

Let $\delta < \alpha + \beta$ be an arbitrary ordinal. We show that $S(\delta) < \alpha + \beta$. We have

$$\delta < \bigcup\{\alpha + \gamma : \gamma < \beta\},$$

so by Proposition 10.2.16, we can fix $\gamma < \beta$ with $\delta < \alpha + \gamma$. Since β is a limit ordinal, we then have $S(\gamma) < \beta$, so

$$\begin{aligned} S(\delta) &\leq \alpha + \gamma \\ &< S(\alpha + \gamma) \\ &= \alpha + S(\gamma) \\ &\leq \alpha + \beta. \end{aligned}$$

It follows that $\alpha + \beta$ is a limit ordinal. □

Proposition 10.3.5. *For all ordinals α, β, and γ, we have $(\alpha + \beta) + \gamma = \alpha + (\beta + \gamma)$.*

Proof. Fix ordinals α and β. We prove that $(\alpha+\beta)+\gamma = \alpha+(\beta+\gamma)$ for all ordinals γ by induction. Suppose first that $\gamma = 0$. We then have

$$\begin{aligned}(\alpha+\beta)+0 &= \alpha+\beta \\ &= \alpha+(\beta+0).\end{aligned}$$

For the successor step, let γ be arbitrary such that $(\alpha+\beta)+\gamma = \alpha+(\beta+\gamma)$. We then have

$$\begin{aligned}(\alpha+\beta)+S(\gamma) &= S((\alpha+\beta)+\gamma) \\ &= S(\alpha+(\beta+\gamma)) \\ &= \alpha+S(\beta+\gamma) \\ &= \alpha+(\beta+S(\gamma)).\end{aligned}$$

For the limit case, let γ be an arbitrary limit ordinal such that $(\alpha+\beta)+\delta = \alpha+(\beta+\delta)$ for all $\delta < \gamma$. We then have

$$\begin{aligned}(\alpha+\beta)+\gamma &= \bigcup\{(\alpha+\beta)+\delta : \delta < \gamma\} \\ &= \bigcup\{\alpha+(\beta+\delta) : \delta < \gamma\} \\ &= \bigcup\{\alpha+\varepsilon : \beta \leq \varepsilon < \beta+\gamma\} \\ &= \bigcup\{\alpha+\varepsilon : \varepsilon < \beta+\gamma\} \\ &= \alpha+(\beta+\gamma),\end{aligned}$$

where the last line follows because $\beta+\gamma$ is a limit ordinal. □

Our recursive definitions of ordinal arithmetic are elegant, but there's an easier way to visualize what they represent.

Proposition 10.3.6. *Let $(W_1, <_1)$ and $(W_2, <_2)$ be well-orderings.*

1. *Let $W = (W_1 \times \{0\}) \cup (W_2 \times \{1\})$, and define a relation $<$ on W as follows.*
   - *For any $v, w \in W_1$, we have $(v, 0) < (w, 0)$ if and only if $v <_1 w$.*
   - *For any $y, z \in W_2$, we have $(y, 1) < (z, 1)$ if and only if $y <_2 z$.*
   - *For any $w \in W_1$ and $z \in W_2$, we have $(w, 0) < (z, 1)$.*

   We then have that $(W, <)$ is well-ordering.
2. *Let $W = W_1 \times W_2$, and define a relation $<$ on W as follows. For any $v, w \in W_1$ and $y, z \in W_2$, we have $(v, y) < (w, z)$ if and only if either $v <_1 w$ or ($v = w$ and $y <_2 z$). We then have that $(W, <)$ is a well-ordering.*

Proof. Exercise (see homework). □

Definition 10.3.7. *Let $(W_1, <_1)$ and $(W_2, <_2)$ be well-orderings.*

1. *We call the ordering $(W, <)$ from (1) above the* sum *of W_1 and W_2 and denote it by $W_1 \oplus W_2$.*
2. *We call the ordering $(W, <)$ from (2) above the* product *of W_1 and W_2 and denote it by $W_1 \otimes W_2$.*

Theorem 10.3.8. *Let α and β be ordinals.*

1. *The well-ordering $\alpha \oplus \beta$ has order-type $\alpha + \beta$.*
2. *The well-ordering $\beta \otimes \alpha$ has order-type $\alpha \cdot \beta$.*

Proof. Exercise. □

10.4 Cardinals

Definition 10.4.1. *A* cardinal *is an ordinal α such that $\alpha \not\approx \beta$ for all $\beta < \alpha$.*

Proposition 10.4.2. *An ordinal α is a cardinal if and only if $\alpha \not\preceq \beta$ for all $\beta < \alpha$.*

Proof. For any $\beta < \alpha$, we trivially have $\beta \preceq \alpha$ because $\beta \subseteq \alpha$. Thus, the result is an immediate consequence of the Cantor-Schröder-Bernstein Theorem. □

Proposition 10.4.3. *Every $n \in \omega$ is a cardinal, and ω is a cardinal.*

Proof. Every $n \in \omega$ is a cardinal by Corollary 9.4.4. Now consider ω. If there existed $n < \omega$ when $\omega \approx n$, then by restricting a witnessing bijection $g\colon \omega \to n$ to the domain $S(n)$, we would obtain an injective function from $S(n)$ to n, contrary to the Pigeonhole Principle. Therefore, ω is a cardinal. □

Proposition 10.4.4. *If κ is a cardinal with $\kappa \not< \omega$, then κ is a limit ordinal.*

Proof. By the homework, we know that $S(\alpha) \approx \alpha$ whenever $\omega \leq \alpha$. Therefore, any successor ordinal greater than or equal to ω must be a limit ordinal. □

Proposition 10.4.5. *Let A be a set. There is an ordinal α such that $\alpha \not\preceq A$.*

Proof. Let $\mathcal{F} = \{(B, R) \in \mathcal{P}(A) \times \mathcal{P}(A \times A) : R \text{ is a well-ordering on } B\}$ be the set of all well-orderings of all subsets of A. By Collection and Separation, the set $T = \{\text{order-type}(B, R) : (B, R) \in \mathcal{F}\}$ is a set of ordinals. Let α be an ordinal such that $\alpha > \bigcup T$ (such an α exists because **ORD** is a proper class).

We claim that $\alpha \not\preceq A$. Suppose instead that $f\colon \alpha \to A$ was injective. Let $B = \text{range}(f)$ and let R be the well-ordering on B obtained by transferring the ordering of α to B via the function f. We would then have that $(B, R) \in \mathcal{F}$ and (B, R) has order-type α, so $\alpha \in T$. This is a contradiction (because $\alpha > \bigcup T$), so $\alpha \not\preceq A$. □

For example, letting $A = \omega$, we conclude that there is an ordinal α such that $\alpha \not\preceq \omega$. In particular, there exists an uncountable ordinal.

Definition 10.4.6. *Let A be a set. The least ordinal α such that $\alpha \not\preceq A$ is called the* Hartogs number *of A, and is denoted by $H(A)$.*

Proposition 10.4.7. *$H(A)$ is a cardinal for every set A.*

Proof. Let A be a set and let $\alpha = H(A)$. Suppose that $\beta < \alpha$ and $\alpha \approx \beta$. Let $f\colon \alpha \to \beta$ be a bijection. Since $\beta < \alpha = H(A)$, there exists an injection $g\colon \beta \to A$. We then have that $g \circ f\colon \alpha \to A$ is an injection, contrary to the fact that $\alpha \not\preceq A$. It follows that $\alpha \not\approx \beta$ for any $\beta < \alpha$, so $H(A) = \alpha$ is a cardinal. □

Definition 10.4.8. *If κ is a cardinal, we let $\kappa^+ = H(\kappa)$.*

Definition 10.4.9. *We define $\aleph_\alpha$ for $\alpha \in$ **ORD** recursively as follows:*

1. $\aleph_0 = \omega$.
2. $\aleph_{\alpha+1} = \aleph_\alpha^+$.
3. $\aleph_\alpha = \bigcup\{\aleph_\beta : \beta < \alpha\}$ *if α is a limit ordinal.*

The following proposition can be proven with a straightforward induction.

Proposition 10.4.10. *Let α and β be ordinals.*

1. $\alpha \leq \aleph_\alpha$.

2. *If $\alpha < \beta$, then $\aleph_\alpha < \aleph_\beta$.*

Proposition 10.4.11. *Let κ be an ordinal. κ is an infinite cardinal if and only if there exists $\alpha \in \boldsymbol{ORD}$ with $\kappa = \aleph_\alpha$.*

Proof. We first prove that $\aleph_\alpha$ is an infinite cardinal for all $\alpha \in \mathbf{ORD}$ by induction. Notice that $\aleph_0 = \omega$ is a cardinal by Proposition 10.4.3. Also, if $\aleph_\alpha$ is a cardinal, then $\aleph_{\alpha+1} = \aleph_\alpha^+ = H(\aleph_\alpha)$ is a cardinal by Proposition 10.4.7. Suppose then that α is a limit ordinal and that $\aleph_\beta$ is a cardinal for all $\beta < \alpha$. Notice that $\aleph_\alpha$ is an ordinal by Proposition 10.2.16. Let $\gamma < \aleph_\alpha$ be arbitrary. Since $\gamma < \aleph_\alpha = \bigcup\{\aleph_\beta : \beta < \alpha\}$, there exists $\beta < \alpha$ such that $\gamma < \aleph_\beta$. Since $\aleph_\beta$ is a cardinal, we know that $\aleph_\beta \not\preceq \gamma$. Now we also have $\aleph_\beta \leq \aleph_\alpha$, so $\aleph_\beta \subseteq \aleph_\alpha$, from which we can conclude that $\aleph_\alpha \not\preceq \gamma$. Therefore $\aleph_\alpha \not\approx \gamma$ for any $\gamma < \aleph_\alpha$, hence $\aleph_\alpha$ is a cardinal.

Conversely, let κ be an arbitrary infinite cardinal. By Proposition 10.4.10, we have $\kappa \leq \aleph_\kappa$. If $\kappa = \aleph_\kappa$, we are done. Suppose then that $\kappa < \aleph_\kappa$. Let α be the least ordinal such that $\kappa < \aleph_\alpha$. Notice that $\alpha \neq 0$ because κ is infinite, and also α can not be a limit ordinal (otherwise, $\kappa < \aleph_\beta$ for some $\beta < \alpha$). Thus, there exists β such that $\alpha = S(\beta)$. By choice of α, we have $\aleph_\beta \leq \kappa$. If $\aleph_\beta < \kappa$, then $\aleph_\beta < \kappa < \aleph_{S(\beta)} = H(\aleph_\beta)$, contradicting the definition of $H(\aleph_\beta)$. It follows that $\kappa = \aleph_\beta$. □

Proposition 10.4.12. *Let A be a set. The following are equivalent:*

1. *There exists an ordinal α such that $A \approx \alpha$.*
2. *A can be well-ordered.*

Proof. Suppose first that there exists an ordinal α such that $A \approx \alpha$. We use a bijection between A and α to transfer the ordering on the ordinals to an ordering on A. Let $f\colon A \to \alpha$ be a bijection. Define a relation $<$ on A by letting $a < b$ if and only if $f(a) < f(b)$. It is then straightforward to check that $(A, <)$ is a well-ordering (using the fact that $(\alpha, \in_\alpha)$ is a well-ordering).

For the converse direction, suppose that A can be well-ordered. Fix a relation $<$ on A so that $(A, <)$ is a well-ordering. By Theorem 10.2.24, there is an ordinal α such that $A \cong \alpha$. In particular, we have $A \approx \alpha$. □

Of course, this leaves open the question of which sets can be well-ordered. Below, we will use the Axiom of Choice to show that every set can be well-ordered.

Definition 10.4.13. *Let A be a set which can be well-ordered. We define $|A|$ to be the least ordinal α such that $A \approx \alpha$.*

Proposition 10.4.14. *If A can be well-ordered, then $|A|$ is a cardinal.*

Proof. Suppose that A can be well-ordered, and let $\alpha = |A|$. Let $\beta < \alpha$ be arbitrary. If $\alpha \approx \beta$, then by composing a bijection from $f\colon A \to \alpha$ with a bijection $g\colon \alpha \to \beta$, we would obtain a bijection from A to β, contradicting the definition of $|A|$. Therefore, $\alpha \not\approx \beta$ for all $\beta < \alpha$, and hence α is a cardinal. □

Given ordinals α and β, we defined the ordinal sum $\alpha + \beta$ and the ordinal product $\alpha \cdot \beta$. Since ordinals are measures of "lengths" of well-orderings, these recursive definitions reflected the "length" of the sum/product of the two well-orderings. In contrast, cardinals are raw measures of "number of elements", not of a length of an ordering of the elements. We now define different notions of *cardinal* addition and multiplication. Let κ and λ be cardinals. Since both $(\kappa \times \{0\}) \cup (\lambda \times \{1\})$ and $\kappa \times \lambda$ can be well-ordered by Proposition 10.3.6, we can make the following definition.

Definition 10.4.15. *Let κ and λ be cardinals. We define the following:*

1. $\kappa + \lambda = |(\kappa \times \{0\}) \cup (\lambda \times \{1\})|$.
2. $\kappa \cdot \lambda = |\kappa \times \lambda|$.

Proposition 10.4.16. *Let κ and λ be cardinals.*

1. $\kappa + \lambda = \lambda + \kappa$.
2. $\kappa \cdot \lambda = \lambda \cdot \kappa$.

Proof. Notice that there is a natural bijection between $(\kappa \times \{0\}) \cup (\lambda \times \{1\})$ and $(\lambda \times \{0\}) \cup (\kappa \times \{1\})$, and there is also a natural bijection between $\kappa \times \lambda$ and $\lambda \times \kappa$. □

Lemma 10.4.17. *Let A_1, A_2, B_1, B_2 be sets with $A_1 \approx A_2$ and $B_1 \approx B_2$.*

1. $(A_1 \times \{0\}) \cup (B_1 \times \{1\}) \approx (A_2 \times \{0\}) \cup (B_2 \times \{1\})$.
2. $A_1 \times B_1 \approx A_2 \times B_2$.

Proof. Exercise. □

The definition of cardinal addition and multiplication is natural, but it is not obvious how to compute many values. Notice that $\aleph_0 + \aleph_0 = \aleph_0$ because $(\aleph_0 \times \{0\}) \cup (\aleph_0 \times \{1\})$ is countable (as the union of two countable sets is countable). Similarly, we have $\aleph_0 \cdot \aleph_0 = \aleph_0$ because $\aleph_0 \times \aleph_0$ is countable (as the Cartesian product of two countable sets is countable). However, what is $\aleph_1 \cdot \aleph_1$? The key to answering this question, and more general questions about cardinal multiplication, is the following important ordering on pairs of ordinals.

Definition 10.4.18. *We define an ordering $<$ on $\boldsymbol{ORD} \times \boldsymbol{ORD}$ as follows. Let $\alpha_1, \beta_1, \alpha_2, \beta_2$ be ordinals. We set $(\alpha_1, \beta_1) < (\alpha_2, \beta_2)$ if one of the following holds.*

1. $\max\{\alpha_1, \beta_1\} < \max\{\alpha_2, \beta_2\}$.
2. $\max\{\alpha_1, \beta_1\} = \max\{\alpha_2, \beta_2\}$ *and* $\alpha_1 < \alpha_2$.
3. $\max\{\alpha_1, \beta_1\} = \max\{\alpha_2, \beta_2\}$, $\alpha_1 = \alpha_2$, *and* $\beta_1 < \beta_2$.

Although this ordering looks strange at first, it fixes several issues with more natural orderings. For example, suppose that we try to order $\mathbf{ORD} \times \mathbf{ORD}$ lexicographically. We could then have that $(0, \alpha) <_{lex} (1, 0)$ for *all* ordinals α, so the class of elements less than $(1, 0)$ is actually a proper class. In contrast, notice that given any $(\alpha, \beta) \in \mathbf{ORD} \times \mathbf{ORD}$, the class

$$\{(\gamma, \delta) \in \mathbf{ORD} \times \mathbf{ORD} : (\gamma, \delta) < (\alpha, \beta)\}$$

is a set, since it is contained in the set $(\max\{\alpha, \beta\} + 1) \times (\max\{\alpha, \beta\} + 1)$. Our ordering is a kind of "graded lexicographic ordering" in that we first order by some kind of "size" (given by the max of the entries), and then order lexicographically inside each "size".

Lemma 10.4.19. *$<$ is a well-ordering on $\boldsymbol{ORD} \times \boldsymbol{ORD}$.*

Proof. Transitivity, asymmetry, and connectedness are easily shown by appealing to the transitivity, asymmetry, and connectedness of the ordering on $\mathbf{ORD}$. Let $\mathbf{C}$ be a nonempty subclass of $\mathbf{ORD} \times \mathbf{ORD}$. Notice that $\mathbf{D} = \{\max\{\alpha, \beta\} : (\alpha, \beta) \in \mathbf{C}\}$ is a nonempty subclass of $\mathbf{ORD}$, hence has a least element δ by Proposition 10.2.18. Now let $A = \{\alpha \in \delta : (\alpha, \delta) \in \mathbf{C}\}$.

Suppose first that $A \neq \emptyset$, and let α_0 be the least element of A (which exists by Proposition 10.2.15). Let $(\alpha, \beta) \in \mathbf{C}$ be arbitrary. Notice that if $\max\{\alpha, \beta\} > \delta$, we then have $(\alpha_0, \delta) < (\alpha, \beta)$. Suppose then that $\max\{\alpha, \beta\} = \delta$. If $\alpha = \delta$, we then have $(\alpha_0, \delta) < (\alpha, \beta)$ because $\alpha_0 < \delta$. If $\alpha \neq \delta$ and $\beta = \delta$, we then have $\alpha_0 \leq \alpha$ by choice of α_0, hence $(\alpha_0, \delta) \leq (\alpha, \beta)$.

Suppose now that $A = \emptyset$. Let $B = \{\beta \in S(\delta) : (\delta, \beta) \in \mathbf{C}\}$ and notice that $B \neq \emptyset$. Let β_0 be the least element of B (which exists by Proposition 10.2.15). Let $(\alpha, \beta) \in \mathbf{C}$. Notice that if $\max\{\alpha, \beta\} > \delta$, we then have $(\delta, \beta_0) < (\alpha, \beta)$. Suppose then that $\max\{\alpha, \beta\} = \delta$. Notice that we must have $\alpha = \delta$ because $A = \emptyset$. It follows that $\beta_0 \leq \beta$ by choice of β_0, hence $(\delta, \beta_0) \leq (\alpha, \beta)$. □

Theorem 10.4.20. *For all $\alpha \in \boldsymbol{ORD}$, we have $\aleph_\alpha \cdot \aleph_\alpha = \aleph_\alpha$.*

Proof. The proof is by induction on $\alpha \in \mathbf{ORD}$. Suppose α is an ordinal and that $\aleph_\beta \cdot \aleph_\beta = \aleph_\beta$ for all $\beta < \alpha$. Notice that if we restrict the $<$ relation on $\mathbf{ORD} \times \mathbf{ORD}$ to $\aleph_\alpha \times \aleph_\alpha$, we still get a well-ordering. Given $(\gamma, \delta) \in \aleph_\alpha \times \aleph_\alpha$, we let

$$P_{\gamma,\delta} = \{(\theta_1, \theta_2) \in \aleph_\alpha \times \aleph_\alpha : (\theta_1, \theta_2) < (\gamma, \delta)\}.$$

Let $(\gamma, \delta) \in \aleph_\alpha \times \aleph_\alpha$ be arbitrary. We claim that $|P_{\gamma,\delta}| < \aleph_\alpha$. To see this, let $\varepsilon = \max\{\gamma, \delta\} + 1$. Now $\aleph_\alpha$ is an infinite cardinal by Proposition 10.4.11, so we know that $\aleph_\alpha$ is a limit ordinal by Proposition 10.4.4. Since $\gamma, \delta < \aleph_\alpha$, it follows that $\varepsilon < \aleph_\alpha$ and hence $|\varepsilon| < \aleph_\alpha$. Now if ε is finite, then $P_{\gamma,\delta}$ is finite, and hence $|P_{\gamma,\delta}| < \aleph_\alpha$ trivially. Otherwise, ε is infinite, and so we can fix $\beta < \alpha$ such that $|\varepsilon| = \aleph_\beta$. We then have $P_{\gamma,\delta} \subseteq \varepsilon \times \varepsilon \approx \aleph_\beta \times \aleph_\beta \approx \aleph_\beta$, by induction, so $|P_{\gamma,\delta}| \leq \aleph_\beta < \aleph_\alpha$. Therefore, $|P_{\gamma,\delta}| < \aleph_\alpha$ for every $(\gamma, \delta) \in \aleph_\alpha \times \aleph_\alpha$.

Since $\aleph_\alpha \times \aleph_\alpha$ is well-ordered by $<$, it follows from Theorem 10.2.24 that $\aleph_\alpha \times \aleph_\alpha \cong \theta$ for some ordinal θ. Let $f : \aleph_\alpha \times \aleph_\alpha \to \theta$ be a witnessing isomorphism. Then f is injective, so we must have $\aleph_\alpha \preceq \theta$, and hence $\aleph_\alpha \leq \theta$. Suppose that $\aleph_\alpha < \theta$. Since f is an isomorphism, there exists $(\gamma, \delta) \in \aleph_\alpha \times \aleph_\alpha$ such that $f((\gamma, \delta)) = \aleph_\alpha$. We then have $|P_{\gamma,\delta}| = \aleph_\alpha$, a contradiction. It follows that $\theta = \aleph_\alpha$, so f witnesses that $\aleph_\alpha \times \aleph_\alpha \approx \aleph_\alpha$. Hence $\aleph_\alpha \cdot \aleph_\alpha = \aleph_\alpha$. □

Corollary 10.4.21. *Suppose that κ and λ are cardinals, $1 \leq \kappa \leq \lambda$, and $\lambda \geq \aleph_0$. We then have*

1. *$\kappa + \lambda = \lambda = \lambda + \kappa$.*
2. *$\kappa \cdot \lambda = \lambda = \lambda \cdot \kappa$.*

Proof. By Proposition 10.4.11, we can fix α such that $\lambda = \aleph_\alpha$. Notice that

$$\kappa \cdot \lambda \leq \lambda \cdot \lambda = \aleph_\alpha \cdot \aleph_\alpha = \aleph_\alpha = \lambda.$$

Since we clearly have $\lambda \leq \kappa \cdot \lambda$, it follows that $\kappa \cdot \lambda = \lambda$. Also, notice that

$$\kappa + \lambda \leq \lambda + \lambda = 2 \cdot \lambda = \lambda,$$

where the last line follows from what we just proved. Since we clearly have $\lambda \leq \kappa + \lambda$, it follows that $\kappa + \lambda = \lambda$. □

10.5 The Axiom of Choice

Definition 10.5.1. *Let $\mathcal{F}$ be a family of nonempty sets. A* choice function *on $\mathcal{F}$ is a function $h : \mathcal{F} \to \bigcup \mathcal{F}$ such that $h(A) \in A$ for all $A \in \mathcal{F}$.*

Proposition 10.5.2. *The following are equivalent (over ZF).*

1. *The Axiom of Choice: If $\mathcal{F}$ is a family of nonempty pairwise disjoint sets, then there is a set C such that $C \cap A$ has a unique element for every $A \in \mathcal{F}$.*
2. *Every family $\mathcal{F}$ of nonempty sets has a choice function.*
3. *Every family $\mathcal{F}$ of nonempty pairwise disjoint sets has a choice function.*

Proof.

- 1 implies 2: Let $\mathcal{F}$ be a family of nonempty sets. Let $\mathcal{G} = \{\{A\} \times A : A \in \mathcal{F}\}$, and notice that $\mathcal{G}$ is a set by Collection and Separation. Furthermore, $\mathcal{G}$ is a family of nonempty pairwise disjoint sets. By 1, there is a set C such that there is unique element of $C \cap B$ for every $B \in \mathcal{G}$. By Separation, we may assume that $C \subseteq \bigcup \mathcal{G}$. Letting $h = C$, it now follows that $h : \mathcal{F} \to \bigcup \mathcal{F}$ and $h(A) \in A$ for every $A \in \mathcal{F}$. Therefore, $\mathcal{F}$ has a choice function.

- 2 implies 3: Trivial.
- 3 implies 1: Let $\mathcal{F}$ be a family of nonempty pairwise disjoint sets. By 3, there is choice function h for $\mathcal{F}$. Let $C = \text{range}(h)$ and notice that there is a unique element of $C \cap A$ for every $A \in \mathcal{F}$ (because the sets in $\mathcal{F}$ are pairwise disjoint).

□

Here are some examples where the Axiom of Choice is implicitly used in mathematics.

Proposition 10.5.3. *If $f\colon A \to B$ is a surjective, there exists an injective $g\colon B \to A$ such that $f \circ g = id_B$.*

The idea of constructing such a g is to let $g(b)$ be an arbitrary $a \in A$ such that $f(a) = b$. When you think about it, there doesn't seem to be a way to define g without making all of these arbitrary choices.

Proof. Define a function $H\colon B \to \mathcal{P}(A)$ by letting $H(b) = \{a \in A : f(a) = b\}$. Notice that $H(b) \neq \emptyset$ for every $b \in B$ because f is surjective. Let $h\colon \mathcal{P}(A)\backslash\{\emptyset\} \to A$ be a choice function, so $h(D) \in D$ for every $D \in \mathcal{P}(A)\backslash\{\emptyset\}$. Set $g = h \circ H$ and notice that $g\colon B \to A$. We first show that $(f \circ g)(b) = b$ for every $b \in B$. Let $b \in B$ be arbitrary. Since $h(H(b)) \in H(b)$, it follows that $f(h(H(b))) = b$, hence

$$\begin{aligned}(f \circ g)(b) &= f(g(b)) \\ &= f(h(H(b))) \\ &= b.\end{aligned}$$

Therefore, $f \circ g$ is the identity function on B. We finally show that g is injective. Let $b_1, b_2 \in B$ be arbitrary with $g(b_1) = g(b_2)$. We then have

$$\begin{aligned}b_1 &= (f \circ g)(b_1) \\ &= f(g(b_1)) \\ &= f(g(b_2)) \\ &= (f \circ g)(b_2) \\ &= b_2.\end{aligned}$$

Therefore, g is injective. □

Proposition 10.5.4. *If $f\colon \mathbb{R} \to \mathbb{R}$ and $y \in \mathbb{R}$, then f is continuous at y if and only if for every sequence $\{x_n\}_{n\in\omega}$ with $\lim\limits_{n\to\infty} x_n = y$, we have $\lim\limits_{n\to\infty} f(x_n) = f(y)$.*

The standard proof of the left-to-right direction makes no use of the Axiom of Choice. For the right-to-left direction, the argument is as follows. Suppose that f is not continuous at y, and fix $\varepsilon > 0$ such that there is no $\delta > 0$ such that whenever $|x - y| < \delta$, we have $|f(x) - f(y)| < \varepsilon$. We define a sequence as follows. Given $n \in \omega$, let x_n be an arbitrary real number with $|x_n - y| < \frac{1}{n}$ such that $|f(x_n) - f(y)| \geq \varepsilon$. Again, we're making infinitely many arbitrary choices in the construction.

Proof. Suppose that f is not continuous at y, and fix $\varepsilon > 0$ such that there is no $\delta > 0$ such that whenever $|x-y| < \delta$, we have $|f(x)-f(y)| < \varepsilon$. Define a function $H\colon \mathbb{R}^+ \to \mathcal{P}(\mathbb{R})$ by letting $H(\delta) = \{x \in \mathbb{R} : |x-y| < \delta$ and $|f(x) - f(y)| \geq \delta$. Notice that $H(\delta) \neq \emptyset$ for every $\delta \in \mathbb{R}^+$ by assumption. Let $h\colon \mathcal{P}(\mathbb{R})\backslash\{\emptyset\} \to \mathbb{R}$ be a choice function. For each $n \in \omega$, let $x_n = h(H(\frac{1}{n}))$. One then easily checks that $\lim\limits_{n\to\infty} x_n = y$ but it's not the case that $\lim\limits_{n\to\infty} f(x_n) = f(y)$. □

Another example is the proof is the countable union of countable sets is countable. Let $\{A_n\}_{n\in\omega}$ be countable sets. The first step is to fix injections $f_n\colon A_n \to \omega$ for each $n \in \omega$ and then build an injection $f\colon \bigcup_{n\in\omega} A_n \to \omega$ from these. However, we are again making infinitely many arbitrary choices when we fix the injections. We'll prove a generalization of this fact using the Axiom of Choice below.

Example. Let $\mathcal{F} = \mathcal{P}(\omega)\backslash\{0\}$. Notice that $\bigcup \mathcal{F} = \omega$. We can prove the existence of a choice function for $\mathcal{F}$ without the Axiom of Choice as follows. Define $g\colon \mathcal{F} \to \omega$ by letting $g(A)$ be the $<$-least element of A for every $A \in \mathcal{P}(\omega)\backslash\{0\}$. More formally, we define $g = \{(A, a) \in \mathcal{F} \times \omega : a \in A \text{ and } a \leq b \text{ for all } b \in A\}$ and prove that g is a choice function on $\mathcal{F}$. □

Proposition 10.5.5. *Without the Axiom of Choice, one can prove that if $\mathcal{F}$ is a family of nonempty sets and $\mathcal{F}$ is finite, then $\mathcal{F}$ has a choice function.*

Theorem 10.5.6 (Zermelo)**.** *The following are equivalent:*

1. *The Axiom of Choice.*
2. *Every set can be well-ordered.*

Proof. 2 implies 1: We show that every family of nonempty sets has a choice function. Let $\mathcal{F}$ be a family of nonempty sets. By 2, we can fix a well-ordering $<$ of $\bigcup \mathcal{F}$. Define $g\colon \mathcal{F} \to \bigcup \mathcal{F}$ by letting $g(A)$ be the $<$-least element of A. Notice that g is a choice function on $\mathcal{F}$.

1 implies 2: Let A be a set. It suffices to show that there is an ordinal α such that $\alpha \approx A$. Since A is a set, we can fix $x \notin A$. Let $g\colon \mathcal{P}(A)\backslash\{\emptyset\} \to A$ be a choice function. We define a class function $\mathbf{F}\colon \mathbf{ORD} \to A \cup \{x\}$ recursively as follows. If $x \in \text{range}(\mathbf{F} \restriction \alpha)$ or $\text{range}(\mathbf{F} \restriction \alpha) = A$, let $\mathbf{F}(\alpha) = x$. Otherwise, $\text{range}(\mathbf{F} \restriction \alpha) \subsetneq A$, and we let $\mathbf{F}(\alpha) = g(A\backslash\text{range}(\mathbf{F} \restriction \alpha))$. Since A is a set and $\mathbf{ORD}$ is a proper class, we know that $\mathbf{F}$ is not injective. It follows that we must have $x \in \text{range}(\mathbf{F})$ (otherwise, a simple induction shows that $\mathbf{F}$ is injective). Let α be the least ordinal such that $\mathbf{F}(\alpha) = x$. A straightforward induction now shows that $\mathbf{F} \restriction \alpha\colon \alpha \to A$ is injective, and we notice that it is surjective because $\mathbf{F}(\alpha) = x$. It follows that $A \approx \alpha$. □

Definition 10.5.7. Zorn's Lemma *is the statement that if $(P, <)$ is nonempty partially ordered set with the property that each chain in P has an upper bound in P, then P has a maximal element.*

Theorem 10.5.8. *The following are equivalent.*

1. *The Axiom of Choice.*
2. *Zorn's Lemma.*

Proof. 1 implies 2: Let $(P, <)$ be nonempty partially ordered set with the property that each chain in P has an upper bound in P. Since P is a set, we can fix $x \notin P$. Let $g\colon \mathcal{P}(P)\backslash\{\emptyset\} \to P$ be a choice function. We define a class function $\mathbf{F}\colon \mathbf{ORD} \to P \cup \{x\}$ recursively as follows. If $x \in \text{range}(\mathbf{F} \restriction \alpha)$, let $\mathbf{F}(\alpha) = x$. Also, if $\text{range}(\mathbf{F} \restriction \alpha) \subseteq P$ and there is no $q \in P$ such that $q > p$ for every $p \in \text{range}(\mathbf{F} \restriction \alpha)$, let $\mathbf{F}(\alpha) = x$. Otherwise, $\text{range}(\mathbf{F} \restriction \alpha) \subseteq P$ and $\{q \in P : q > p \text{ for every } p \in \text{range}(\mathbf{F} \restriction \alpha)\} \neq \emptyset$, and we let $\mathbf{F}(\alpha) = g(\{q \in P : q > p$ for every $p \in \text{range}(\mathbf{F} \restriction \alpha)\})$. We know that $\mathbf{F}$ can not be injective, so as above we must have $x \in \text{range}(\mathbf{F})$. Fix the least ordinal α such that $\mathbf{F}(\alpha) = x$. A straightforward induction shows that $\text{range}(\mathbf{F} \restriction \alpha)$ is injective and that $\text{range}(\mathbf{F} \restriction \alpha)$ is a chain in P.

Notice that $\alpha \neq 0$ because $P \neq \emptyset$. Suppose that α is a limit ordinal. Since $\text{range}(\mathbf{F} \restriction \alpha)$ is a chain in P, we know by assumption that there exists $q \in P$ with $q \geq p$ for all $p \in \text{range}(\mathbf{F} \restriction \alpha)$. Notice that we can not have $q = \mathbf{F}(\beta)$ for any $\beta < \alpha$ because we would then have $\beta + 1 < \alpha$ (because α is a limit ordinal) and $q < \mathbf{F}(\beta + 1)$ by definition of $\mathbf{F}$, contrary to the fact that $q \geq p$ for all $p \in \text{range}(\mathbf{F} \restriction \alpha)$. It follows that $q > p$ for all $p \in \text{range}(\mathbf{F} \restriction \alpha)$, hence $\mathbf{F}(\alpha) \neq x$, a contradiction. It follows that α is a successor ordinal, say $\alpha = S(\beta)$. Since $\mathbf{F}(\beta) \neq x$ and $\mathbf{F}(S(\beta)) = x$, it follows that $\mathbf{F}(\beta)$ is a maximal element of P.

2 implies 1: Let $\mathcal{F}$ be a family of nonempty sets. We use Zorn's Lemma to show that $\mathcal{F}$ has a choice function. Let $P = \{q : q \text{ is a function, } \text{domain}(q) \subseteq \mathcal{F}, \text{ and } q(A) \in A \text{ for every } A \in \text{domain}(q)\}$. Given $p, q \in P$, we let $p < q$ if and only if $p \subsetneq q$. It is easy to check that $(P, <)$ is a partial ordering. Notice that $P \neq \emptyset$ because $\emptyset \in P$. Also, if H is a chain in P, then $\bigcup H \in P$, and $p \leq \bigcup H$ for all $p \in H$. It follows that every chain in P has an upper bound in P. By Zorn's Lemma, P has a maximal element which we call g. We

need only show that $\text{domain}(g) = \mathcal{F}$. Suppose instead that $\text{domain}(g) \subsetneq \mathcal{F}$, and fix $A \in \mathcal{F} \backslash \text{domain}(g)$. Fix $a \in A$. We then have $g \cup \{(A, a)\} \in P$ and $g < g \cup \{(A, a)\}$, a contradiction. It follows that $\text{domain}(g) = \mathcal{F}$, so g is a choice function on $\mathcal{F}$. □

Definition 10.5.9. *Let V be a vector space over a field F, and let $S \subseteq V$. We define*

$$Span_F(S) = \left\{ \sum_{i=1}^{n} \lambda_i w_i : n \in \mathbb{N}, \lambda_i \in F, w_i \in S \right\}.$$

In other words, $Span_F(S)$ is the set of all finite *linear combinations of elements of S.*

Definition 10.5.10. *Let V be a vector space over a field F, and let $S \subseteq V$.*

- *We say that S is* linearly independent *if for all* distinct $w_1, w_2, \ldots, w_n \in S$ *and all* $\lambda_1, \lambda_2, \ldots, \lambda_n \in F$ *with* $\lambda_1 w_1 + \lambda_2 w_2 + \cdots + \lambda_n w_n = 0$, *we have* $\lambda_i = 0$ *for all* i.
- *We say that S is a basis of V if S is linearly independent and $Span_F(S) = V$.*

Proposition 10.5.11. *Let S be a linearly independent subset of V, and let $v \in V \backslash S$. The following are equivalent:*

1. *$v \notin Span_F(S)$.*
2. *$S \cup \{v\}$ is linearly independent.*

Proof. We prove the contrapositive of each direction. Suppose first that $v \in Span_F(S)$. Fix distinct $w_1, w_2, \ldots, w_n \in S$ and all $\lambda_1, \lambda_2, \ldots, \lambda_n \in F$ with $\lambda_1 w_1 + \lambda_2 w_2 + \cdots + \lambda_n w_n = v$. We then have $(-1)v + \lambda_1 w_1 + \lambda_2 w_2 + \cdots + \lambda_n w_n = 0$, so since $-1 \neq 0$, we conclude that $S \cup \{v\}$ is linearly dependent.

Conversely, suppose that $S \cup \{v\}$ is linearly dependent. Fix $w_1, w_2, \ldots, w_n \in V$, and $\mu, \lambda_1, \lambda_2, \ldots, \lambda \in F$, at least one of which is nonzero, with $\mu v + \lambda_1 w_1 + \lambda_2 w_2 + \cdots + \lambda_n w_n = 0$. Since S is linearly independent, we must have $\mu \neq 0$. We then have

$$v = \left(\frac{-\lambda_1}{\mu} \right) w_1 + \left(\frac{-\lambda_2}{\mu} \right) w_2 + \cdots + \left(\frac{-\lambda_n}{\mu} \right) w_n,$$

so $v \in Span_F(S)$. □

Theorem 10.5.12. *If V is a vector space over F, then there exists a basis of V.*

Proof. The key fact is that if $\mathcal{G}$ is a set of linearly independent subsets of V that is linearly ordered by $\subseteq$ (i.e. for all $S_1, S_2 \in \mathcal{G}$, either $S_1 \subseteq S_2$ or $S_2 \subseteq S_1$), then $\bigcup \mathcal{G}$ is linearly independent. To see this, let $w_1, w_2, \ldots, w_n \in \bigcup \mathcal{G}$ and $\lambda_1, \lambda_2, \ldots, \lambda_n \in F$ be arbitrary with $\lambda_1 w_1 + \lambda_2 w_2 + \cdots + \lambda_n w_n = 0$. For each i, fix $S_i \in \bigcup \mathcal{G}_i$ with $w_i \in S_i$. Now the set $\mathcal{G}$ is linearly ordered with respect to $\subseteq$, so we can fix k with $1 \leq k \leq n$ such that $S_i \subseteq S_k$ for all i. We then have $w_i \in S_k$ for all i, so since S_k is linearly independent and $\lambda_1 w_1 + \lambda_2 w_2 + \cdots + \lambda_n w_n = 0$, we conclude that $\lambda_i = 0$ for all i. Therefore, $\bigcup \mathcal{G}$ is linearly independent.

We now apply either Zorn's Lemma on the set of linearly independent subsets of V ordered by inclusion, or use transfinite induction (taking unions of limit ordinals), to obtain a linearly independent set S that can not be extended to another linearly independent set. Using Proposition 10.5.11, we conclude that $Span_F(S) = V$. Therefore, S is a basis for V. □

Once we adopt the Axiom of Choice, it follows that every set can be well-ordered. Therefore, $|A|$ is defined for every set A.

Proposition 10.5.13. *Let A and B be sets.*

1. *$A \preceq B$ if and only if $|A| \leq |B|$.*
2. *$A \approx B$ if and only if $|A| = |B|$.*

Proof.

1. Suppose first that $|A| \leq |B|$. Let $\kappa = |A|$ and let $\lambda = |B|$, and fix bijections $f : A \to \kappa$ and $g : \lambda \to B$. Since $\kappa \leq \lambda$, we have $\kappa \subseteq \lambda$ and so we may consider $g \circ f : A \to B$. One easily checks that this is an injective function, so $A \preceq B$.

   Suppose now that $A \preceq B$, and fix an injection $h : A \to B$. Let $\kappa = |A|$ and let $\lambda = |B|$, and fix bijections $f : \kappa \to A$ and $g : B \to \lambda$. We then have that $g \circ h \circ f : \kappa \to \lambda$ is injective, so $\kappa \preceq \lambda$. Since κ is cardinal, we know from Proposition 10.4.2 that $\lambda \not\approx \kappa$, so $\kappa \leq \lambda$.

2. Suppose first that $A \approx B$. We then have that $|A| \leq |B|$ and $|B| \leq |A|$ by part 1, hence $|A| = |B|$.

   Suppose now that $|A| = |B|$. Let κ be this common value, and fix bijections $f : A \to \kappa$ and $g : \kappa \to B$. We then have that $g \circ f : A \to B$ is a bijection, so $A \approx B$.

□

Proposition 10.5.14. *$|A \times A| = |A|$ for every infinite set A.*

Proof. Since A is infinite, we can fix an ordinal α with $|A| = \aleph_\alpha$ by Proposition 10.4.11. We then have $A \times A \approx \aleph_\alpha \times \aleph_\alpha \approx \aleph_\alpha$ by Theorem 10.4.20, so $|A \times A| = \aleph_\alpha$. □

Proposition 10.5.15. *Let $\mathcal{F}$ be a family of sets. Suppose that $|\mathcal{F}| \leq \kappa$ and that $|A| \leq \lambda$ for every $A \in \mathcal{F}$. We then have $|\bigcup \mathcal{F}| \leq \kappa \cdot \lambda$.*

Proof. Let $\mu = |\mathcal{F}|$ (notice that $\mu \leq \kappa$), and fix a bijection $f : \mu \to \mathcal{F}$. Also, for each $A \in \mathcal{F}$, fix an injection $g_A : A \to \lambda$ (using the Axiom of Choice). Define a function $h : \bigcup \mathcal{F} \to \kappa \times \lambda$ as follows. Given $b \in \bigcup \mathcal{F}$, let α be the least ordinal such that $b \in f(\alpha)$, and set $h(b) = (\alpha, g_{f(\alpha)}(b))$.

We claim that h is injective. Let $b_1, b_2 \in \bigcup \mathcal{F}$ be arbitrary with $h(b_1) = h(b_2)$. Let α_1 be the least ordinal such that $b_1 \in f(\alpha_1)$ and let α_2 be the least ordinal such that $b_2 \in f(\alpha_2)$. Since $h(b_1) = h(b_2)$, it follows that $\alpha_1 = \alpha_2$, and we call their common value α. Therefore, using the fact that $h(b_1) = h(b_2)$ again, we conclude that $g_{f(\alpha)}(b_1) = g_{f(\alpha)}(b_2)$. Since $g_{f(\alpha)}$ is an injection, it follows that $b_1 = b_2$. Hence, $h : \mathcal{F} \to \kappa \times \lambda$ is injective, so we may conclude that $|\mathcal{F}| \leq \kappa \cdot \lambda$. □

Proposition 10.5.16. *$|A^{<\omega}| = |A|$ for every infinite set A.*

Proof. Using Proposition 10.5.14 and induction (on ω), it follows that $|A^n| = |A|$ for every $n \in \omega$ with $n \geq 1$. Since $A^{<\omega} = \bigcup\{A^n : n \in \omega\}$, we may use Proposition 10.5.15 to conclude that $|A^{<\omega}| \leq \aleph_0 \cdot |A| = |A|$. We clearly have $|A| \leq |A^{<\omega}|$, hence $|A^{<\omega}| = |A|$. □

Definition 10.5.17. *Let A and B be sets. We let A^B be the set of all functions from B to A.*

Proposition 10.5.18. *Let A_1, A_2, B_1, B_2 be sets with $A_1 \approx A_2$ and $B_1 \approx B_2$. We then have $A_1^{B_1} \approx A_2^{B_2}$.*

Proof. Exercise. □

Now that we've adopted the Axiom of Choice, we know that A^B can be well-ordered for any sets A and B, so it makes sense to talk about $|A^B|$. This gives us a way to define cardinal exponentiation.

Definition 10.5.19. *Let κ and λ be cardinals. We use κ^λ to also denote the cardinality of the set κ^λ. (So, we're using the same notation κ^λ to denote both the set of functions from λ to κ and also its cardinality).*

Proposition 10.5.20. *Let κ, λ, and μ be cardinals.*

1. $\kappa^{\lambda+\mu} = \kappa^\lambda \cdot \kappa^\mu$.
2. $\kappa^{\lambda\cdot\mu} = (\kappa^\lambda)^\mu$.
3. $(\kappa \cdot \lambda)^\mu = \kappa^\mu \cdot \lambda^\mu$.

Proof. Fix sets A, B, C such that $|A| = \kappa$, $|B| = \lambda$, and $|C| = \mu$ (we could just use κ, λ, and μ, but it's easier to distinguish sets from cardinals).

1. It suffices to find a bijection $F\colon A^{B\times\{0\}\cup C\times\{1\}} \to A^B \times A^C$. We define F as follows. Given $f\colon B \times \{0\} \cup C \times \{1\} \to A$, let $F(f) = (g, h)$ where $g\colon B \to A$ is given by $g(b) = f((b, 0))$ and $h\colon C \to A$ is given by $h(c) = f((c, 1))$.
2. It suffices to find a bijection $F\colon (A^B)^C \to A^{B\times C}$. We define F as follows. Given $f\colon C \to A^B$, let $F(f)\colon B \times C \to A$ be the function defined by $F(f)((b, c)) = f(c)(b)$ for all $b \in B$ and $c \in C$.
3. It suffices to find a bijection $F\colon A^C \times B^C \to (A \times B)^C$. We define F as follows. Given $g\colon C \to A$ and $h\colon C \to B$, let $F((g, h))\colon C \to A \times B$ be the function defined by $F((g, h))(c) = (g(c), h(c))$ for all $c \in C$.

In each case, it is straightforward to check the given F is indeed a bijection. □

Proposition 10.5.21. *$2^\kappa = |\mathcal{P}(\kappa)|$ for all cardinals κ.*

Proof. Let κ be an arbitrary cardinal. We define a function $F\colon 2^\kappa \to \mathcal{P}(\kappa)$ as follows. Given $f\colon \kappa \to 2$, let $F(f) = \{\alpha \in \kappa : f(\alpha) = 1\}$. We then have that F is a bijection, hence $2^\kappa = |\mathcal{P}(\kappa)|$. □

Corollary 10.5.22. *$\kappa < 2^\kappa$ for all cardinals κ.*

Proof. We know that $\kappa \prec \mathcal{P}(\kappa)$ from Theorem 9.6.6. □

Proposition 10.5.23. *If $2 \leq \lambda \leq \kappa$, then $\lambda^\kappa = 2^\kappa$*

Proof. Notice that

$$2^\kappa \leq \lambda^\kappa \leq \kappa^\kappa \leq (2^\kappa)^\kappa = 2^{\kappa\cdot\kappa} = 2^\kappa,$$

so we must have $\lambda^\kappa = 2^\kappa$. □

10.6 Exercises

1. Let $(W_1, <_1)$ and $(W_2, <_2)$ be well-orderings.
   (a) Let $W = (W_1 \times \{0\}) \cup (W_2 \times \{1\})$, and define a relation $<$ on W as follows:
   - For any $v, w \in W_1$, we have $(v, 0) < (w, 0)$ if and only if $v <_1 w$.
   - For any $y, z \in W_2$, we have $(y, 1) < (z, 1)$ if and only if $y <_2 z$.
   - For any $w \in W_1$ and $z \in W_2$, we have $(w, 0) < (z, 1)$.

   Show that $(W, <)$ is a well-ordering. We call W the *sum* of W_1 and W_2 and denote it by $W_1 \oplus W_2$.
   (b) Let $W = W_1 \times W_2$, and define a relation $<$ on W as follows. For any $v, w \in W_1$ and $y, z \in W_2$, we have $(v, y) < (w, z)$ if and only if either $v <_1 w$ or ($v = w$ and $y <_2 z$). Show that $(W, <)$ is a well-ordering. We call W the *product* of W_1 and W_2 and denote it by $W_1 \otimes W_2$.
2. In this problem, α, β, and γ are always ordinals.
   (a) Show that if $\alpha \leq \beta$, there exists a unique γ such that $\alpha + \gamma = \beta$.

(b) Give examples of $\alpha \leq \beta$ such that the equation $\gamma + \alpha = \beta$ has 0, 1, and infinitely many solutions for γ.

3. Show that if α is an infinite ordinal, then $S(\alpha) \approx \alpha$.

4. Let $A \subseteq \mathbb{R}$. Suppose that $(A, <)$ is a well-ordering under the usual ordering $<$ on $\mathbb{R}$. Show that A is countable.

   Hint: Define an injective function from A to $\mathbb{Q}$.

5. Show that for every set A, there exists a transitive set T with the following properties:

   - $A \subseteq T$.
   - $T \subseteq S$ for all transitive sets S with $A \subseteq S$.

   T is called the *transitive closure* of A.

6. Define a transfinite sequence of sets V_α for $\alpha \in \mathbf{ORD}$ by:

   - $V_0 = \emptyset$.
   - $V_{\alpha+1} = \mathcal{P}(V_\alpha)$ for all ordinals α.
   - $V_\alpha = \bigcup\{V_\beta : \beta < \alpha\}$ for all limit ordinals α.

   (a) Show that V_α is transitive for all ordinals α.

   (b) Show that if $\beta < \alpha$, then $V_\beta \subseteq V_\alpha$.

   (c) Show that if $x, y \in V_\omega$, then $\bigcup x$, $\{x, y\}$, and $\mathcal{P}(x)$ are all elements of V_ω.

   Note: Working in ZFC without Foundation, one can show that the Axiom of Foundation is equivalent to the statement that for every set x, there exists an ordinal α with $x \in V_\alpha$.

7. Give a careful proof (meaning that you'll use the Axiom of Choice so be explicit when using it) of the following fact: A linear ordering $(L, <)$ is a well-ordering if and only if there is no $f : \omega \to L$ such that $f(n+1) < f(n)$ for all $n \in \omega$.

8. Let $\mathcal{L} = \{\mathsf{R}\}$ where R is a binary relation symbol. Show that the class of well-orderings is not a weak elementary class in the language $\mathcal{L}$.

9. Over ZF, show that the statement "For all sets A and B and all surjections $f : A \to B$, there exists an injection $g : B \to A$ such that $f \circ g$ is the identity function on B" implies the Axiom of Choice.

10. The Hausdorff Maximality Principle states that every partial ordering $(P, <)$ has a maximal chain with respect to $\subseteq$ (that is, a chain H such that there is no chain I with $H \subsetneq I$). Show that the Hausdorff Maximality Principle is equivalent to the Axiom of Choice over ZF.

11. (**) Show that for every ordinal α, there exists ordinals $\beta_1, \beta_2, \ldots, \beta_k$ with $\alpha \geq \beta_1 > \beta_2 > \cdots > \beta_k$ and $n_1, n_2, \ldots, n_k \in \omega$ such that

$$\alpha = \omega^{\beta_1} \cdot n_1 + \omega^{\beta_2} \cdot n_2 + \cdots + \omega^{\beta_k} \cdot n_k.$$

    Furthermore, show that the β_i and n_i in such a representation are unique. This expression of an ordinal in "base ω" is called *Cantor Normal Form*.

Chapter 11

Set-theoretic Methods in Mathematics

11.1 Subsets of $\mathbb{R}$

In this section, we try to understand the real numbers using set-theoretic tools. The first connection is expressing $|\mathbb{R}|$ in terms of cardinal exponentiation.

Proposition 11.1.1. $|\mathbb{R}| = 2^{\aleph_0}$.

Proof. The function $f\colon \mathbb{R} \to \mathcal{P}(\mathbb{Q})$ given by $f(x) = \{q \in \mathbb{Q} : q < x\}$ is injective (because $\mathbb{Q}$ is dense in $\mathbb{R}$), so

$$|\mathbb{R}| \leq |\mathcal{P}(\mathbb{Q})| = 2^{|\mathbb{Q}|} = 2^{\aleph_0}.$$

Now let $\mathcal{F}$ be the set of all functions from ω to 2. The function $f\colon \mathcal{F} \to \mathbb{R}$ defined by

$$f(q) = \sum_{n=0}^{\infty} \frac{q(n)}{10^{n+1}}$$

is injective (by simple properties of decimal expansions). Therefore, $2^{\aleph_0} = |\mathcal{F}| \leq |\mathbb{R}|$. □

Although interesting, we have not yet determined $2^{\aleph_0}$. That is, we know that $2^{\aleph_0} = \aleph_\alpha$ for some $\alpha \in \mathbf{ORD}$, but we do not know the value of α. Since $2^{\aleph_0} = |\mathcal{P}(\omega)|$, we know that $\aleph_0 < 2^{\aleph_0}$, so $\alpha > 0$. A natural guess is that $2^{\aleph_0}$ is the first uncountable cardinal $\aleph_1$. This guess is called the *Continuum Hypothesis*. One way to attack this problem is to try to show that for every $A \subseteq \mathbb{R}$, either A is countable or $A \approx \mathbb{R}$. If successful, then we could conclude that every cardinal strictly less than $2^{\aleph_0}$ is countable, which would solve the Continuum Hypothesis affirmatively. We start by analyzing the simplest types of subsets of $\mathbb{R}$.

Proposition 11.1.2. *If $a, b \in \mathbb{R}$ and $a < b$, then $|(a, b)| = 2^{\aleph_0}$.*

Proof. If $\mathcal{F}$ is the set of all functions from ω to 2, then the function $f\colon \mathcal{F} \to (0, 1)$ defined by

$$f(q) = \sum_{n=0}^{\infty} \frac{q(n) + 1}{10^{n+1}}$$

is injective (by simple properties of decimal expansions as above). Thus, $|(0, 1)| \geq 2^{\aleph_0}$. Since $(0, 1) \subseteq \mathbb{R}$, we also have $|(0, 1)| \leq 2^{\aleph_0}$, so $|(0, 1)| = 2^{\aleph_0}$.

Now given any $a, b \in \mathbb{R}$ with $a < b$, we have $(0, 1) \approx (a, b)$ via the function $f(x) = a + x \cdot (b - a)$, so $|(a, b)| = |(0, 1)| = 2^{\aleph_0}$. □

Proposition 11.1.3. *If O is a nonempty open subset of $\mathbb{R}$, then $|O| = 2^{\aleph_0}$.*

Proof. Every nonempty open subset of $\mathbb{R}$ contains an open interval. □

Now that we have handled open sets, we move on to closed sets. We start with the following special types of closed sets.

Definition 11.1.4. *Let $P \subseteq \mathbb{R}$. We say that P is* perfect *if it is closed and has no isolated points.*

For example, the closed interval $[a, b]$ is perfect for all $a, b \in \mathbb{R}$ with $a < b$. A more interesting example is given by the Cantor set defined by

$$C = \left\{ \sum_{n=0}^{\infty} \frac{q(n)}{3^{n+1}} : q \in \{0, 2\}^{\omega} \right\}.$$

Consult your favorite analysis book to see that C is perfect.

Proposition 11.1.5. *If $P \subseteq \mathbb{R}$ is perfect and $a, b \in \mathbb{R}$ with $a < b$ and $a, b \notin P$, then $P \cap [a, b]$ is perfect.*

Proof. Since both P and $[a, b]$ are closed, it follows that $P \cap [a, b]$ is closed. Let $x \in P \cap [a, b]$ be arbitrary, and notice that $x > a$ and $x < b$ since $a, b \notin P$. Let $\varepsilon > 0$. Since P is perfect, we know that x is not isolated in P, so there exists $y \in P$ such that $0 < |x - y| < \min\{\varepsilon, x - a, b - x\}$. We then have that $0 < |x - y| < \varepsilon$ and also that $y \in [a, b]$ (by choice of ε). Therefore, x is not isolated in $P \cap [a, b]$. It follows that $P \cap [a, b]$ is perfect. □

Definition 11.1.6. *Let $A \subseteq \mathbb{R}$. We define $diam(A) = \sup\{|x - y| : x, y \in A\}$.*

Proposition 11.1.7. *If $P \subseteq \mathbb{R}$ is a nonempty perfect set and $\varepsilon > 0$, then there exists nonempty perfect sets $P_1, P_2 \subseteq \mathbb{R}$ with the following properties:*

1. *$P_1 \cap P_2 = \emptyset$.*
2. *$P_1 \cup P_2 \subseteq P$.*
3. *$diam(P_1), diam(P_2) < \varepsilon$.*

Proof. Let $P \subseteq \mathbb{R}$ be a nonempty perfect set and let $\varepsilon > 0$. Since P is nonempty, we may fix $x \in P$.

- *Case 1:* There exists $\delta > 0$ such that $[x - \delta, x + \delta] \subseteq P$. We may assume (by making δ smaller if necessary) that $\delta < \varepsilon$. In this case, let $P_1 = [x - \delta, x - \frac{\delta}{2}]$ and let $P_2 = [x + \frac{\delta}{2}, x + \delta]$.
- *Case 2:* Otherwise, for every $\delta > 0$, there exists infinitely many $y \in [x - \delta, x + \delta] \backslash P$. Thus, there exists points $a, b, c, d \in [x - \frac{\varepsilon}{4}, x + \frac{\varepsilon}{4}] \backslash P$ such that $a < b < c < d$. In this case, let $P_1 = P \cap [a, b]$ and let $P_2 = P \cap [c, d]$.

□

Proposition 11.1.8. *If $P \subseteq \mathbb{R}$ is a nonempty perfect set, then $|P| = 2^{\aleph_0}$.*

Proof. Since $P \subseteq \mathbb{R}$, we know that $|P| \leq 2^{\aleph_0}$. By Proposition 11.1.7, there exists a nonempty perfect set $Q \subseteq P$ such that $diam(Q) < 1$. We can now use Proposition 11.1.7 to recursively define a function $f \colon 2^{<\omega} \to \mathcal{P}(P)$ with the following properties:

1. $f(\lambda) = Q$.
2. $f(\sigma)$ is a nonempty perfect set for all $\sigma \in 2^{<\omega}$.

3. $diam(f(\sigma)) < \frac{1}{2^{|\sigma|}}$ for all $\sigma \in 2^{<\omega}$.
4. $f(\sigma * 0) \cup f(\sigma * 1) \subseteq f(\sigma)$ for all $\sigma \in 2^{<\omega}$.
5. $f(\sigma * 0) \cap f(\sigma * 1) = \emptyset$ for all $\sigma \in 2^{<\omega}$.

Now define $g\colon 2^{\aleph_0} \to P$ by letting $g(q)$ be the unique element of $\bigcap_{n\in\omega} f(q \restriction n)$ for all $q \in 2^{\aleph_0}$ (notice that such an element must exist because the intersection is of a nested sequence of compact sets, and that the element is unique because the diameters go to 0). Finally, notice that g is injective by virtue of property 5 of the function f. □

Definition 11.1.9. *Suppose that $C \subseteq \mathbb{R}$ is a closed set. We define*

$$C' = C \backslash \{x \in \mathbb{R} : x \text{ is an isolated point of } C\}.$$

We call C' the Cantor-Bendixson derivative *of C.*

Notice that a closed set C is perfect if and only if $C = C'$.

Proposition 11.1.10. *If $C \subseteq \mathbb{R}$ is a closed set, then $C' \subseteq C$ is also closed.*

Proof. Recall that a set is closed if and only if its complement is open. We show that $\overline{C'}$ is open. Let $x \in \overline{C'}$ be arbitrary. If $x \notin C$, then since C is closed, we may fix $\delta > 0$ such that $(x - \delta, x + \delta) \subseteq \overline{C} \subseteq \overline{C'}$. Suppose then that $x \in C$. Since $x \notin C'$, we know that x is an isolated point of C. Fix $\delta > 0$ such that $C \cap (x - \delta, x + \delta) = \{x\}$. We then have that $(x - \delta, x + \delta) \subseteq \overline{C'}$. Therefore, $\overline{C'}$ is open. It follows that C' is closed. □

Proposition 11.1.11. *If $C \subseteq \mathbb{R}$ is a closed set, then $C \backslash C' = \{x \in \mathbb{R} : x \text{ is an isolated point of } C\}$ is countable.*

Proof. Define a function $f\colon C \backslash C' \to \mathbb{Q} \times \mathbb{Q}$ by letting $f(x) = (q, r)$ where (q, r) is least (under some fixed well-ordering of $\mathbb{Q} \times \mathbb{Q}$) such that $C \cap (q, r) = \{x\}$. We then have that f is injective, hence $C \backslash C'$ is countable because $\mathbb{Q} \times \mathbb{Q}$ is countable. □

In attempting to find a perfect set inside of a closed set, we begin by throwing out the isolated points. However, there might be new isolated points once we throw out the original ones. Thus, we may have to repeat this process. In fact, we may have to repeat it beyond ω. Let ω_1 be the first uncountable ordinal (i.e. $\omega_1 = \aleph_1$, but thought of as an ordinal rather than a cardinal).

Definition 11.1.12. *Let $C \subseteq \mathbb{R}$ be a closed set. We define a sequence $C^{(\alpha)}$ for $\alpha < \omega_1$ recursively as follows:*

1. $C^{(0)} = C$.
2. $C^{(\alpha+1)} = (C^{(\alpha)})'$.
3. $C^{(\alpha)} = \bigcap \{C^{(\beta)} : \beta < \alpha\}$ *if α is a limit.*

Notice that each $C^{(\alpha)}$ is closed and that $C^{(\beta)} \subseteq C^{(\alpha)}$ whenever $\alpha < \beta < \omega_1$ by a trivial induction.

Proposition 11.1.13. *Let $C \subseteq \mathbb{R}$ be a closed set. There exists an $\alpha < \omega_1$ such that $C^{(\alpha+1)} = C^{(\alpha)}$.*

Proof. Suppose that $C^{(\alpha+1)} \neq C^{(\alpha)}$ for all $\alpha < \omega_1$. Define a function $f\colon \omega_1 \to \mathbb{Q} \times \mathbb{Q}$ by letting $f(\alpha) = (q, r)$ where (q, r) is least (under some fixed well-ordering of $\mathbb{Q} \times \mathbb{Q}$) such that there is a unique element of $C^{(\alpha)} \cap (q, r)$. We then have that f is injective, contrary to the fact that $|\mathbb{Q} \times \mathbb{Q}| = \aleph_0$. □

Theorem 11.1.14. *Let $C \subseteq \mathbb{R}$ be a closed set. There exists a perfect set $P \subseteq \mathbb{R}$ and a countable $A \subseteq \mathbb{R}$ such that $C = A \cup P$ and $A \cap P = \emptyset$.*

Proof. Fix $\alpha < \omega_1$ such that $C^{(\alpha+1)} = C^{(\alpha)}$. Let $P = C^{(\alpha)}$ and let $A = \bigcup_{\beta<\alpha}(C^{(\beta)} \backslash C^{(\beta+1)})$. Notice that $C = A \cup P$ and $A \cap P = \emptyset$. Furthermore, P is perfect because $P = P'$, and A is countable because it is the countable union of countable sets. □

Corollary 11.1.15. *If $C \subseteq \mathbb{R}$ is an uncountable closed set, then $|C| = 2^{\aleph_0}$.*

Proof. Let $C \subseteq \mathbb{R}$ be an uncountable closed set. We have $|C| \leq 2^{\aleph_0}$ because $C \subseteq \mathbb{R}$. Let P be perfect and A countable such that $C = A \cup P$ and $A \cap P = \emptyset$. Since C is uncountable, we have $P \neq \emptyset$, hence $|P| = 2^{\aleph_0}$, and so $|C| \geq 2^{\aleph_0}$. □

In order to move on, we need to discuss more complicated types of subsets of $\mathbb{R}$. The next most natural class of sets are the Borel sets.

Definition 11.1.16. *Let $\mathcal{O}$ be the set of open subsets of $\mathbb{R}$. We define the set $\mathcal{B}$ of* Borel sets *to be the smallest subset of $\mathcal{P}(\mathbb{R})$ such that*

1. *$\mathcal{O} \subseteq \mathcal{B}$.*
2. *If $A \in \mathcal{B}$, then $\mathbb{R} \backslash A \in \mathcal{B}$.*
3. *If $A_n \in \mathcal{B}$ for all $n \in \omega$, then $\bigcup_{n\in\omega} A_n \in \mathcal{B}$.*

It turns out that every Borel set is either countable or has size $2^{\aleph_0}$. However, this is harder to prove. Also, there are subsets of $\mathbb{R}$ that are not Borel, and it quickly becomes difficult to get a handle on these more "pathological" sets. The study of Borel sets and higher generalizations (such as analytic and projective sets) is part of a subject called *descriptive set theory.*

11.2 The Size of Models

Theorem 11.2.1 (Downward Lowenheim-Skolem-Tarski Theorem)**.** *Suppose that $\mathcal{L}$ is a language with $|\mathcal{L}| \leq \kappa$ (i.e. $|\mathcal{C} \cup \mathcal{R} \cup \mathcal{F}| \leq \kappa$), that $\mathcal{M}$ is an $\mathcal{L}$-structure, and that $X \subseteq M$ is such that $|X| \leq \kappa$. There exists $\mathcal{A} \preceq \mathcal{M}$ such that $X \subseteq A$ and $|A| \leq \kappa$.*

Proof. Follow the proof of Theorem 4.5.4, with an appropriate analogue of Problem 6 on Homework 1. □

Corollary 11.2.2. *Let $\mathcal{L}$ be a language and suppose that $\Gamma \subseteq Form_{\mathcal{L}}$ is satisfiable. There exists a model $(\mathcal{M}, s)$ of Γ such that $|M| \leq |\mathcal{L}| + \aleph_0$.*

Proof. Since Γ is satisfiable, we can fix a model $(\mathcal{N}, s)$ of Γ. Let $X = \text{range}(s)$, By the Downward Lowenheim-Skolem-Tarski Theorem, we can fix $\mathcal{M} \preceq \mathcal{N}$ with $\text{range}(s) \subseteq M$ and $|M| \leq |\mathcal{L}| + \aleph_0$. □

Theorem 11.2.3 (Lowenheim-Skolem Theorem)**.** *Let $\mathcal{L}$ be a language and suppose that $\Gamma \subseteq Form_{\mathcal{L}}$ has an infinite model. Let $\kappa \geq |\mathcal{L}| + \aleph_0$. There exists a model $(\mathcal{M}, s)$ of Γ such that $|M| = \kappa$.*

Proof. Suppose that $\kappa \geq |\mathcal{L}|$. Let $\mathcal{L}'$ be $\mathcal{L}$ together with new constant symbols c_α for all $\alpha < \kappa$. Notice that $|\mathcal{L}'| = |\mathcal{L}| + \kappa = \kappa$. Let

$$\Gamma' = \Gamma \cup \{\mathsf{c}_\alpha \neq \mathsf{c}_\beta : \alpha, \beta < \kappa \text{ and } \alpha \neq \beta\}$$

Notice that every finite subset of Γ' has a model by using an infinite model of Γ and interpreting the constants which appear in the finite subset as distinct elements. Therefore, by Compactness, we know that Γ is satisfiable. By Corollary 11.2.2, there exists a model $(\mathcal{M}', s)$ of Γ' such that $|M'| \leq |\mathcal{L}'| + \aleph_0 = \kappa$. Notice that we must also have $|M'| \geq \kappa$, hence $|M'| = \kappa$. Letting $\mathcal{M}$ be the restriction of the structure $\mathcal{M}'$ to the language $\mathcal{L}$, we see that $(\mathcal{M}, s)$ is a model of Γ and that $|M| = \kappa$. □

Definition 11.2.4. *Given a theory T in a language $\mathcal{L}$ and a cardinal κ, let $I(T,\kappa)$ be the number of models of T of cardinality κ up to isomorphism.*

Proposition 11.2.5. *Let T be a theory in a language $\mathcal{L}$ with $|\mathcal{L}| = \lambda$. For any infinite cardinal κ, we have $I(T,\kappa) \leq 2^{\kappa\cdot\lambda}$. In particular, if $\kappa \geq \lambda$ is infinite, then $I(T,\kappa) \leq 2^{\kappa}$.*

Proof. Let κ be an infinite cardinal. We have

$$\begin{aligned} I(T,\kappa) &\leq \kappa^{|\mathcal{C}|} \cdot |\mathcal{P}(\kappa^{<\omega})|^{|\mathcal{R}|} \cdot |\mathcal{P}(\kappa^{<\omega})|^{|\mathcal{F}|} \\ &\leq \kappa^{|\mathcal{C}|} \cdot |\mathcal{P}(\kappa)|^{|\mathcal{R}|} \cdot |\mathcal{P}(\kappa)|^{|\mathcal{F}|} \\ &\leq \kappa^{\lambda} \cdot (2^{\kappa})^{\lambda} \cdot (2^{\kappa})^{\lambda} \\ &\leq (2^{\kappa})^{\lambda} \cdot (2^{\kappa})^{\lambda} \cdot (2^{\kappa})^{\lambda} \\ &= 2^{\kappa\cdot\lambda} \end{aligned}$$

Now if $\kappa \geq \lambda$, we have $\kappa \cdot \lambda = \kappa$, so $I(T,\kappa) \leq 2^{\kappa}$. □

Proposition 11.2.6. *If T is the theory of groups, then $I(T,\aleph_0) = 2^{\aleph_0}$.*

Proof. Let P be the set of primes. Notice that the set of finite subsets of P is countable, so the set of infinite subsets of P has cardinality $2^{\aleph_0}$. For each infinite $A \in \mathcal{P}(P)$, let

$$G_A = \oplus_{p\in A}\mathbb{Z}/p\mathbb{Z}.$$

In other words, G_A is the set of all functions f with domain A with the following properties:

- $f(p) \in \mathbb{Z}/p\mathbb{Z}$ for each $p \in A$.
- $\{p \in A : f(p) \neq 0\}$ is finite.

Notice that G_A is countable for each infinite $A \in \mathcal{P}(P)$. Now if $A, B \in \mathcal{P}(P)$ are both infinite with $A \neq B$, then $G_A \ncong G_B$, because if $A \nsubseteq B$, say, and we fix $p \in A\backslash B$, then G_A has an element of order p but G_B does not. □

Proposition 11.2.7. *Let T be the theory of vector spaces over $\mathbb{Q}$. We have $I(T,\aleph_0) = \aleph_0$ and $I(T,\kappa) = 1$ for all $\kappa \geq \aleph_1$.*

Proof. Notice first that if V is a vector space over $\mathbb{Q}$ and $dim_{\mathbb{Q}}(V) = n \in \omega$, then

$$|V| = |\mathbb{Q}^n| = \aleph_0.$$

Now if V is a vector space over $\mathbb{Q}$ and $dim_{\mathbb{Q}}(V) = \kappa \geq \aleph_0$, then since every element of V is a finite sum of scaler multiples of elements of a basis, it follows that

$$|V| \leq |(\mathbb{Q}\times\kappa)^{<\omega}| = |(\aleph_0\cdot\kappa)^{<\omega}| = |\kappa^{<\omega}| = \kappa.$$

and we clearly have $|V| \geq \kappa$, so $|V| = \kappa$.

Since two vector spaces over $\mathbb{Q}$ are isomorphic if and only if they have the same dimension, it follows that $I(T,\aleph_0) = \aleph_0$ (corresponding to dimensions in $\omega \cup \{\aleph_0\}$) and $I(T,\kappa) = 1$ for all $\kappa \geq \aleph_1$ (corresponding to dimension κ). □

Definition 11.2.8. *Let F and K be fields with $F \subseteq K$. Given a set $S \subseteq K$, we say that S is* algebraically independent *over F if whenever $p(x_1,\ldots,x_n) \in F[x_1,\ldots,x_n]$ is a polynomial, and $a_1,\ldots,a_n \in S$ are distinct with $p(a_1,\ldots,a_n) = 0$, we have $p = 0$.*

In field theory, there is an analogue of dimension that is called *transcendence degree*. While the dimension of a vector space is the cardinality of maximal linearly independent sets, the transcendence degree of a field extension is the cardinality of maximal algebraically independent sets. Following the above outline, it is possible to prove that two algebraic closed fields of a fixed characteristic are isomorphic if and only if they have the same transcendence degree over their prime subfield. This leads to the following result.

Definition 11.2.9. *Let F and K be fields with $F \subseteq K$. A set $S \subseteq K$ is called a* transcendence base *of K over F if S is algebraically independent over F, and K is algebraic over $F(S)$.*

Proposition 11.2.10. *Let F and K be fields with $F \subseteq K$. Let $S \subseteq K$ be algebraically independent. If $b \in K$ is transcendental over $F(S)$, then $S \cup \{b\}$ is algebraically independent over F.*

Proof. Let $p(x_1, \ldots, x_n, y) \in F[x_1, \ldots, x_n, y]$ and distinct $a_1, \ldots, a_n \in S$ be arbitrary with $p(a_1, \ldots, a_n, b) = 0$. Write $p(x_1, \ldots, x_n, y)$ as a polynomial in y with coefficients in $F[x_1, \ldots, x_n, y]$, so

$$p(x_1, \ldots, x_n, y) = q_k(x_1, \ldots, x_n) \cdot y^k + q_{k-1}(x_1, \ldots, x_n) \cdot y^{k-1} + \cdots + q_0(x_1, \ldots, x_n)$$

where $q_i(x_1, \ldots, x_n) \in F[x_1, \ldots, x_n]$ for all i. We then have

$$q_k(a_1, \ldots, a_n) \cdot b^k + q_{k-1}(a_1, \ldots, a_n) \cdot b^{k-1} + \cdots + q_0(a_1, \ldots, a_n) = 0.$$

Now each of the coefficients is an element of $F(S)$, so the above shows b as a root of a polynomial over $F(S)$. Since b is transcendental over $F(S)$, the polynomial must be 0, so we must have $q_i(a_1, \ldots, a_n) = 0$, so since S is algebraically independent over F, it follows that $q_i = 0$ for all i. Therefore, $p = 0$. □

Proposition 11.2.11. *Let F and K be fields with $F \subseteq K$. There exists a transcendence base of K over F*

Proof. Iterate above result through the ordinals (or use Zorn's Lemma). □

Proposition 11.2.12. *Let F and K be fields with $F \subseteq K$. If K is algebraic over F, then $|K| \leq \max\{|F|, \aleph_0\}$.*

Proof. Define a function $g\colon F^* \to F[x]$ by letting $g(a_0 a_1 a_2 \ldots a_n)$ be the polynomial $a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n$. We then trivially have that g is surjective (although not injective, since $g(2, 1, 0) = 2 + x = g(2, 1)$). Thus $|F[x]| \leq |F^*| = \max\{|F|, \aleph_0\}$.

For each nonzero $p(x) \in F[x]$, let $R_{p(x)}$ be the set of roots of $p(x)$ in K, i.e. let $R_{p(x)} = \{a \in K : p(a) = 0\}$. Now we know from algebra that a nonzero polynomial $p(x)$ of degree n has at most n roots. In particular, for every nonzero $p(x) \in F[x]$, the set $R_{p(x)}$ is finite, and hence countable. Since K is algebraic over F, we have

$$K = \bigcup_{p(x) \in F[x] \setminus \{0\}} R_{p(x)}.$$

Using the fact that $|F[x]\setminus\{0\}| = \max\{|F|, \aleph_0\}$, it follows that $|K| \leq |F[x]| \cdot \aleph_0 = \max\{|F|, \aleph_0\}$. □

Proposition 11.2.13. *Let F be a field and let I be a nonempty set. Let R be the polynomial ring over F in indeterminates indexed by I, and let K be the fraction field of R We then have that $|K| = |R| = \max\{|F|, \aleph_0\}$.*

Proof. Let J be the set of functions from I to $\mathbb{N}$ such that only finitely many elements of I are sent to nonzero values.

The ring R is the set of functions from I to F with finite support □

Theorem 11.2.14. *For any p, we have $I(ACF_p, \aleph_0) = \aleph_0$ and $I(ACF_p, \kappa) = 1$ for all $\kappa \geq \aleph_1$.*

Definition 11.2.15. *Let T be a theory and let κ be a cardinal. We say that T is* κ-categorical *if $I(T, \kappa) = 1$.*

Proposition 11.2.16 (Łos-Vaught Test)**.** *Suppose that T is a theory such that all models of T are infinite. If there exists $\kappa \geq |\mathcal{L}| + \aleph_0$ such that T is κ-categorical, then T is complete.*

Proof. Let T be a theory such that all models of T are infinite. We prove the contrapositive. Suppose that T is not complete and fix $\sigma \in Sent_{\mathcal{L}}$ such that $\sigma \notin T$ and $\neg\sigma \notin T$. We then have that $T \cup \{\sigma\}$ and $T \cup \{\neg\sigma\}$ are both satisfiable with infinite models (because all models of T are infinite), so by the Lowenheim-Skolem Theorem we may fix a model $\mathcal{M}_1$ of $T \cup \{\sigma\}$ and a model $\mathcal{M}_2$ of $T \cup \{\neg\sigma\}$ such that $|M_1| = \kappa = |M_2|$. We then have that $\mathcal{M}_1$ and $\mathcal{M}_2$ are models of T which are not isomorphic, hence $I(T, \kappa) \geq 2$, and so T is not κ-categorical. □

Corollary 11.2.17. *If T is the theory of vector spaces over $\mathbb{Q}$, then T is complete.*

Corollary 11.2.18. *ACF_0 is complete, and each ACF_p is complete.*

The Lowenheim-Skolem says that a first-order theory is unable to control the cardinalities of the infinite models (since as soon as there is one infinite model, there is an infinite model of every cardinality greater than or equal to $|\mathcal{L}|$). However, there are some surprising limitations on the *number* of models of various cardinalities. For example, we have the following result.

Theorem 11.2.19 (Morley's Theorem)**.** *Let $\mathcal{L}$ be a countable language and let T be a theory. If T is κ-categorical for some $\kappa \geq \aleph_1$, then T is κ-categorical for all $\kappa \geq \aleph_1$.*

Morley's Theorem is quite deep, and marks the beginning of modern model theory.

11.3 Ultraproducts and Compactness

Let $\mathcal{L}$ be a language, let I be a set, and suppose that for each $i \in I$ we have an $\mathcal{L}$-structure $\mathcal{M}_i$. For initial clarity, think of the case where $I = \omega$, so we have $\mathcal{L}$-structures $\mathcal{M}_0$, $\mathcal{M}_1$, $\mathcal{M}_2$, ... We want a way to put together all of the $\mathcal{M}_i$ which "blends" the properties of the $\mathcal{M}_i$ together into one structure. An initial thought is to form a product of the structures $\mathcal{M}_i$ with underlying set $\prod_{i\in I} M_i$. That is, M consists of all functions $g\colon I \to \bigcup_{i\in I} M_i$ such that $g(i) \in M_i$ for all $i \in I$. Interpreting the constants and functions would then be straightforward.

For example, suppose that $\mathcal{L} = \{\mathsf{e}, \mathsf{f}\}$ where e is a constant symbol and f is a binary relation symbol. Suppose that $I = \omega$ and that each $\mathcal{M}_i$ is a group. Elements of M would then be sequences $\langle a_i\rangle_{i\in\omega}$, we would interpret e as the sequence of each identity in each group, and we would interpret f as the componentwise group operation (i.e. $f^{\mathcal{M}}(\langle a_i\rangle_{i\in\omega}, \langle b_i\rangle_{i\in\omega}) = \langle f^{\mathcal{M}_i}(a_i, b_i)\rangle_{i\in\omega}$. For a general set I and language without any relation symbols, we would let $\mathsf{c}^{\mathcal{M}}$ be the function $i \mapsto \mathsf{c}^{\mathcal{M}_i}$ for each constant symbol c, and given $\mathsf{f} \in \mathcal{F}_k$ we would let $\mathsf{f}^{\mathcal{M}_i}(g_1, g_2, \ldots, g_k)$ be the function $i \mapsto \mathsf{f}^{\mathcal{M}_i}(g_i(i), g_2(i), \ldots, g_k(i))$.

This certainly works, but it doesn't really "blend" the properties of the structures together particularly well. For example, if each $\mathcal{M}_i$ is a group and all but one is abelian, the product is still nonabelian. Also, if we have relation symbols, it's not clear what the "right" way to determine how to interpret the relation on $\mathcal{M}$. For example, if $\mathcal{L} = \{\mathsf{R}\}$ where R is a binary relation symbol and $I = \omega$, do we say that the pair $(\langle a_i\rangle_{i\in\omega}, \langle b_i\rangle_{i\in\omega})$ is an element of $\mathsf{R}^{\mathcal{M}}$ if *some* $(a_i, b_i) \in \mathsf{R}^{\mathcal{M}_i}$, if *all* $(a_i, b_i) \in \mathsf{R}^{\mathcal{M}_i}$, or something else? Which is the "right" definition? In other words, if each $\mathcal{M}_i$ is a graph, do we put an edge between the sequences if some edge exists between the components, or if every pair has an edge?

To give a uniformly suitable answer to these questions, we want a more "democratic" approach of forming $\mathcal{M}$ that also gives a way to nicely interpret the relation symbols. If I were finite, perhaps we could do a majority rules (if *most* of the pairs were in the relation), but what if I is infinite?

Definition 11.3.1. *Let X be a set. A* filter *on X is a set $\mathcal{F} \subseteq \mathcal{P}(X)$ such that*

1. *$X \in \mathcal{F}$ and $\emptyset \notin \mathcal{F}$.*
2. *If $A \in \mathcal{F}$ and $A \subseteq B \subseteq X$, then $B \in \mathcal{F}$.*

3. $A \cap B \in \mathcal{F}$ *whenever* $A, B \in \mathcal{F}$.

Example. Let X be a nonempty set, and let $x \in X$. The set

$$\mathcal{F} = \{A \in \mathcal{P}(X) : x \in A\}$$

is a filter on X. Such a filter is called a *principal* filter on X generated by x. □

Proposition 11.3.2. *Let X be an infinite set. The set*

$$\mathcal{F} = \{A \in \mathcal{P}(X) : A \text{ is cofinite}\}$$

is a filter on X.

Proof. Immediate from the fact that the intersection of two cofinite sets is cofinite. □

Proposition 11.3.3. *Let X be a set and let $\mathcal{F}$ be a filter on X. For every finite $\mathcal{T} \subseteq \mathcal{F}$, we have $\bigcap \mathcal{T} \in \mathcal{F}$. In particular, for every finite $\mathcal{T} \subseteq \mathcal{F}$, we have $\bigcap \mathcal{T} \neq \emptyset$.*

Proof. A straightforward induction on $|\mathcal{T}|$. □

Definition 11.3.4. *Let X be a set and suppose that $\mathcal{S} \subseteq \mathcal{P}(X)$. We say that $\mathcal{S}$ has the* finite intersection property *if $\bigcap \mathcal{T} \neq \emptyset$ for all finite $\mathcal{T} \subseteq \mathcal{S}$.*

Proposition 11.3.5. *Let X be a set and suppose that $\mathcal{S} \subseteq \mathcal{P}(X)$. The following are equivalent*

1. *$\mathcal{S}$ has the finite intersection property.*
2. *There exists a filter $\mathcal{F}$ on X such that $\mathcal{S} \subseteq \mathcal{F}$.*

Proof. 1 implies 2: Let

$$\mathcal{F} = \{A \in \mathcal{P}(X) : \bigcap \mathcal{T} \subseteq A \text{ for some finite } \mathcal{T} \subseteq \mathcal{S}\}$$

We claim that $\mathcal{F}$ is a filter on X. Notice that we clearly have $X \in \mathcal{F}$, and that $\emptyset \notin \mathcal{F}$ because $\mathcal{S}$ has the finite intersection property. Now if $A \in \mathcal{F}$, say $\bigcap \mathcal{T} \subseteq A$ where $\mathcal{T} \subseteq \mathcal{S}$ is finite, and $A \subseteq B \subseteq X$, then $\bigcap \mathcal{T} \subseteq B$, so $B \in \mathcal{F}$. Finally, suppose that $A, B \in \mathcal{F}$, and fix finite $\mathcal{T}_1, \mathcal{T}_2 \subseteq \mathcal{S}$ such that $\bigcap \mathcal{T}_1 \subseteq A$ and $\bigcap \mathcal{T}_2 \subseteq B$. We then have that $\bigcap(\mathcal{T}_1 \cup \mathcal{T}_2) \subseteq A \cap B$, hence $A \cap B \in \mathcal{F}$.

2 implies 1: Fix a filter $\mathcal{F}$ on X with $\mathcal{S} \subseteq \mathcal{F}$. Let $\mathcal{T}$ be a finite subset of $\mathcal{S}$. Using Proposition 11.3.3, we can immediately conclude that $\bigcap \mathcal{T} \neq \emptyset$. □

Definition 11.3.6. *Let X be a set. An* ultrafilter *on X is filter $\mathcal{U}$ on X such that for all $A \subseteq X$, either $A \in \mathcal{U}$ or $X \backslash A \in \mathcal{U}$.*

For example, every principal filter is an ultrafilter (because given $x \in X$, we have that for all $A \in \mathcal{P}(X)$, either $x \in A$ or $x \in X \backslash A$).

Proposition 11.3.7. *Let $\mathcal{F}$ be a filter on X. $\mathcal{F}$ is an ultrafilter on X if and only if $\mathcal{F}$ is a maximal filter on X (i.e. there is no filter $\mathcal{G}$ on X with $\mathcal{F} \subsetneq \mathcal{G}$).*

Proof. Suppose that $\mathcal{F}$ is not a maximal filter on X. Fix a filter $\mathcal{G}$ on X such that $\mathcal{F} \subsetneq \mathcal{G}$. Fix $A \in \mathcal{G} \backslash \mathcal{F}$. Notice that $X \backslash A \notin \mathcal{F}$ because otherwise we would have $X \backslash A \in \mathcal{G}$ and hence $\emptyset = A \cap (X \backslash A) \in \mathcal{G}$, a contradiction. Therefore, $A \notin \mathcal{F}$ and $X \backslash A \notin \mathcal{F}$, so $\mathcal{F}$ is not an ultrafilter on X.

Conversely, suppose that $\mathcal{F}$ is not an ultrafilter on X. Fix $A \in \mathcal{P}(X)$ such that $A \notin \mathcal{F}$ and $X \backslash A \notin \mathcal{F}$. We claim that $\mathcal{F} \cup \{A\}$ has the finite intersection property. To see this, suppose that $B_1, B_2, \ldots, B_n \in \mathcal{F}$. We then have $B_1 \cap B_2 \cap \cdots \cap B_n \in \mathcal{F}$, so $B_1 \cap B_2 \cap \cdots \cap B_n \neq \emptyset$. Furthermore, since $B_1 \cap B_2 \cap \cdots \cap B_n \in \mathcal{F}$ and $X \backslash A \notin \mathcal{F}$, we must have $B_1 \cap B_2 \cap \cdots \cap B_n \nsubseteq X \backslash A$, so $B_1 \cap B_2 \cap \cdots \cap B_n \cap A \neq \emptyset$. Therefore, $\mathcal{F} \cup \{A\}$ has the finite intersection property. Using Proposition 11.3.5, we can fix a filter $\mathcal{G}$ on X such that $\mathcal{F} \cup \{A\} \subseteq \mathcal{G}$. Since $\mathcal{F} \subsetneq \mathcal{G}$ (as $A \in \mathcal{G} \backslash \mathcal{F}$), it follows that $\mathcal{F}$ is not a maximal filter on X. □

Proposition 11.3.8. *Let $\mathcal{F}$ be a filter on X. There exists an ultrafilter $\mathcal{U}$ on X such that $\mathcal{F} \subseteq \mathcal{U}$.*

Proof. Apply Zorn's Lemma, using the fact that a union of a chain of filters on X is a filter on X. □

Corollary 11.3.9. *Let X be an infinite set. There exists a nonprincipal ultrafilter on X.*

Proof. Let $\mathcal{F}$ be the filter on X consisting of all cofinite subsets of X. Fix an ultrafilter $\mathcal{U}$ on X such that $\mathcal{F} \subseteq \mathcal{U}$. For all $x \in X$, we have $X \backslash \{x\} \in \mathcal{F} \subseteq \mathcal{U}$, hence $\{x\} \notin \mathcal{U}$. □

Ultrafilters (or even just filters) solve our democratic blending problem for relation symbols beautifully. Suppose that $\mathcal{L} = \{\mathsf{R}\}$ where R is a binary relation symbol and $I = \omega$. Suppose also that $\mathcal{U}$ is an ultrafilter on ω. Given elements $\langle a_i \rangle_{i \in \omega}$ and $\langle b_i \rangle_{i \in \omega}$ of M, we could then say that the pair $(\langle a_i \rangle_{i \in \omega}, \langle b_i \rangle_{i \in \omega})$ is an element of $\mathsf{R}^{\mathcal{M}}$ if the set of indices $i \in I$ such that $(a_i, b_i) \in \mathsf{R}^{\mathcal{M}_i}$ is "large", i.e. if $\{i \in I : (a_i, b_i) \in \mathsf{R}^{\mathcal{M}_i}\} \in \mathcal{U}$. Of course, our notion of "large" depends on the ultrafilter, but that flexibility is the beauty of the construction!

However, we have yet to solve the dictatorial problem of function symbols (such as the product of groups in which each is abelian save one ending up nonabelian regardless of what we consider "large"). Wonderfully, and perhaps surprisingly, the ultrafilter can be used in another way to save the day. For concreteness, consider the situation where $\mathcal{L} = \{\mathsf{e}, \mathsf{f}\}$ where e is a constant symbol and f is a binary relation symbol, $I = \omega$, and each $\mathcal{M}_i$ is a group. The idea is to flat out ignore variations on "small" sets by considering two sequences $\langle a_i \rangle_{i \in \omega}$ and $\langle b_i \rangle_{i \in \omega}$ to be the same if the set of indices in which they agree is "large", i.e. if $\{i \in I : a_i = b_i\} \in \mathcal{U}$. In other words, we should define an equivalence relation $\sim$ in this way and take a quotient! This is completely analagous to considering two functions $f, g \colon \mathbb{R} \to \mathbb{R}$ to be the same if the set $\{x \in \mathbb{R} : f(x) \neq g(x)\}$ has measure 0. What does this solve? Suppose that $\mathcal{M}_0$ was our rogue nonabelian group, and each $\mathcal{M}_i$ for $i \neq 0$ was an abelian group. Suppose also that $\omega \backslash \{0\} \in \mathcal{U}$ (i.e. our ultrafilter is not the principal ultrafilter generated by $\{0\}$, and thus we are considering $\{0\}$ to be a "small" set). Given a sequence $\langle a_i \rangle_{i \in \omega}$, let $[\langle a_i \rangle_{i \in \omega}]$ be the equivalence class of $\langle a_i \rangle_{i \in \omega}$ under the relation. Assuming that everything is well-defined (see below), we then have that $\langle \mathsf{f}^{\mathcal{M}_i}(a_i, b_i) \rangle_{i \in \omega} \sim \langle \mathsf{f}^{\mathcal{M}_i}(b_i, a_i) \rangle_{i \in \omega}$ and so

$$\begin{aligned} \mathsf{f}^{\mathcal{M}}([\langle a_i \rangle_{i \in \omega}], [\langle b_i \rangle_{i \in \omega}]) &= [\langle \mathsf{f}^{\mathcal{M}_i}(a_i, b_i) \rangle_{i \in \omega}] \\ &= [\langle \mathsf{f}^{\mathcal{M}_i}(b_i, a_i) \rangle_{i \in \omega}] \\ &= \mathsf{f}^{\mathcal{M}}([\langle b_i \rangle_{i \in \omega}], [\langle a_i \rangle_{i \in \omega}]) \end{aligned}$$

and so we have saved abelianess by ignoring problems on "small" sets!

To summarize before launching into details, here's the construction. Start with a language $\mathcal{L}$, a set I, and $\mathcal{L}$-structures $\mathcal{M}_i$ for each $i \in I$. Form the product $\prod_{i \in I} M_i$, but take a quotient by considering two elements of this product to be equivalent if the set of indices on which they agree is "large". Elements of our structure are now equivalence classes, so we need to worry about things being well-defined, but the fundamental idea is to interpret constant symbols and functions componentwise, and interpret relation symbols by saying that that an k-tuple is in the interpretation of some $\mathsf{R} \in \mathcal{R}_k$ if the set of indices on which the corresponding k-tuple is in $\mathsf{R}^{\mathcal{M}_i}$ is "large". Amazingly, this process behaves absolutely beautifully with regards to first-order logic. For example, if we denote this "blended" structure by $\mathcal{M}$, we will prove below that for any $\sigma \in Sent_{\mathcal{L}}$ we have

$$\mathcal{M} \vDash \sigma \text{ if and only if } \{i \in I : \mathcal{M}_i \vDash \sigma\} \in \mathcal{U}.$$

That is, an arbitrary sentence σ is true in the "blended" structure if and only if the set of indices $i \in I$ in which σ is true in $\mathcal{M}_i$ is "large"!

Onward to the details. The notation is painful and easy to get lost in, but keep the fundamental ideas in mind and revert to thinking of $I = \omega$ whenever the situation looks hopelessly complicated. First we have the proposition saying that the $\sim$ defined in this way is an equivalence relation and that our definitions are well-defined.

Proposition 11.3.10. *Let I be a set, and suppose that for each $i \in I$ we have an $\mathcal{L}$-structure $\mathcal{M}_i$. Let $\mathcal{U}$ be an ultrafilter on I. Define a relation $\sim$ on $\prod_{i\in I} M_i$ by saying that $g \sim h$ if $\{i \in I : g(i) = h(i)\} \in \mathcal{U}$.*

1. *$\sim$ is an equivalence relation on $\prod_{i\in I} M_i$.*
2. *Suppose that $g_1, g_2, \ldots, g_k, h_1, h_2, \ldots, h_k \in \prod_{i\in I} M_i$ are such that $g_j \sim h_j$ for all j.*
   (a) *$\{i \in I : (g_1(i), g_2(i), \ldots, g_k(i)) = (h_1(i), h_2(i), \ldots, h_k(i))\} \in \mathcal{U}$.*
   (b) *For each $\mathsf{R} \in \mathcal{R}_k$, the following are equivalent:*
      - *$\{i \in I : (g_1(i), g_2(i), \ldots, g_k(i)) \in \mathsf{R}^{\mathcal{M}_i}\} \in \mathcal{U}$.*
      - *$\{i \in I : (h_1(i), h_2(i), \ldots, h_k(i)) \in \mathsf{R}^{\mathcal{M}_i}\} \in \mathcal{U}$.*
   (c) *For each $\mathsf{f} \in \mathcal{F}_k$, we have $\{i \in I : \mathsf{f}^{\mathcal{M}_i}(g_1(i), g_2(i), \ldots, g_k(i)) = \mathsf{f}^{\mathcal{M}_i}(h_1(i), h_2(i), \ldots, h_k(i))\} \in \mathcal{U}$.*

Proof. Exercise. □

Definition 11.3.11. *Let I be a set, and suppose that for each $i \in I$ we have an $\mathcal{L}$-structure $\mathcal{M}_i$. Let $\mathcal{U}$ be an ultrafilter on I. We define an $\mathcal{L}$-structure $\mathcal{M} = \prod_{i\in I} \mathcal{M}_i/\mathcal{U}$ as follows. Define the relation $\sim$ on $\prod_{i\in I} \mathcal{M}_i$ by saying that $g \sim h$ if $\{i \in I : g(i) = h(i)\} \in \mathcal{U}$ (as above), and let the universe of $\mathcal{M}$ be the corresponding quotient (i.e. set of equivalence classes). Interpret the symbols of $\mathcal{L}$ as follows:*

1. *For each $\mathsf{c} \in \mathcal{C}$, let $\mathsf{c}^{\mathcal{M}} = [i \mapsto \mathsf{c}^{\mathcal{M}_i}]$.*
2. *For each $\mathsf{R} \in \mathcal{R}_k$, let $\mathsf{R}^{\mathcal{M}} = \{([g_1], [g_2], \ldots, [g_k]) \in M^k : \{i \in I : (g_1(i), g_2(i), \ldots, g_k(i)) \in \mathsf{R}^{\mathcal{M}_i}\} \in \mathcal{U}\}$.*
3. *For each $\mathsf{f} \in \mathcal{F}_k$, let $\mathsf{f}^{\mathcal{M}}([g_1], [g_2], \ldots, [g_k]) = [i \mapsto \mathsf{f}^{\mathcal{M}_i}(g_1(i), g_2(i), \ldots, g_k(i))]$.*

We call $\mathcal{M}$ the ultraproduct *of the $\mathcal{M}_i$ over the ultrafilter $\mathcal{U}$.*

Definition 11.3.12. *In the above situation, given variable assignments $s_i \colon Var \to M_i$ for each $i \in I$, we let $\langle s_i \rangle_{i\in I}$ denote the variable assigment $Var \to M$ given by $\langle s_i \rangle_{i\in I}(\mathsf{x}) = [i \mapsto s_i(\mathsf{x})]$.*

Lemma 11.3.13. *Let $\mathcal{L}$ be a language, let I be a set, and let $\mathcal{U}$ be an ultrafilter on I. Suppose that for each $i \in I$, we have an $\mathcal{L}$-structure $\mathcal{M}_i$, and let $\mathcal{M} = \prod_{i\in I} \mathcal{M}_i/\mathcal{U}$. For all $t \in Term_{\mathcal{L}}$ and all $s_i \colon Var \to M_i$, we have*

$$\overline{\langle s_i \rangle_{i\in I}}(t) = [i \mapsto \overline{s_i}(t)]$$

In other words, for all $t(\mathsf{x}_1, \mathsf{x}_2, \ldots, \mathsf{x}_k) \in Term_{\mathcal{L}}$ and all $g_1, g_2, \ldots, g_k \in \prod_{i\in I} M_i$, we have

$$t^{\mathcal{M}}([g_1], [g_2], \ldots, [g_k]) = [i \mapsto t^{\mathcal{M}_i}(g_1(i), g_2(i), \ldots, g_k(i))]$$

Proof. Suppose that $\mathsf{c} \in \mathcal{C}$. Let $s_i \colon Var \to M_i$ be variable assignments. We then have

$$\begin{aligned} \overline{\langle s_i \rangle_{i\in I}}(\mathsf{c}) &= \mathsf{c}^{\mathcal{M}} \\ &= [i \mapsto \mathsf{c}^{\mathcal{M}_i}] \\ &= [i \mapsto \overline{s_i}(\mathsf{c})] \end{aligned}$$

Suppose that $\mathsf{x} \in Var$. Let $s_i \colon Var \to M_i$ be variable assignments. We then have

$$\begin{aligned} \overline{\langle s_i \rangle_{i\in I}}(x) &= \langle s_i \rangle_{i\in I}(\mathsf{x}) \\ &= [i \mapsto s_i(\mathsf{x})] \\ &= [i \mapsto \overline{s_i}(\mathsf{x})] \end{aligned}$$

Suppose that $\mathsf{f} \in \mathcal{F}_k$ and $t_1, t_2, \ldots, t_k \in Term_{\mathcal{L}}$ are such that the result holds for the t_i. Let $s_i \colon Var \to M_i$ be variable assignments. We then have

$$\begin{aligned}
\overline{\langle s_i \rangle_{i \in I}}(\mathsf{f} t_1 t_2 \cdots t_k) &= \mathsf{f}^{\mathcal{M}}(\overline{\langle s_i \rangle_{i \in I}}(t_1), \overline{\langle s_i \rangle_{i \in I}}(t_2), \ldots, \overline{\langle s_i \rangle_{i \in I}}(t_k)) \\
&= \mathsf{f}^{\mathcal{M}}([i \mapsto \overline{s_i}(t_1)], [i \mapsto \overline{s_i}(t_2)], \ldots, [i \mapsto \overline{s_i}(t_k)]) \\
&= [i \mapsto \mathsf{f}^{\mathcal{M}_i}(\overline{s_i}(t_1), \overline{s_i}(t_2), \ldots, \overline{s_i}(t_k))] \\
&= [i \mapsto \overline{s_i}(\mathsf{f} t_1 t_2 \cdots t_k)]
\end{aligned}$$

□

Theorem 11.3.14 (Łos)**.** *Let $\mathcal{L}$ be a language, let I be a set, and let $\mathcal{U}$ be an ultrafilter on I. Suppose that for each $i \in I$, we have an $\mathcal{L}$-structure $\mathcal{M}_i$, and let $\mathcal{M} = \prod_{i \in I} \mathcal{M}_i / \mathcal{U}$. For all $\varphi \in Form_{\mathcal{L}}$ and all $s_i \colon Var \to M_i$, we have*

$$(\mathcal{M}, \langle s_i \rangle_{i \in I}) \vDash \varphi \text{ if and only if } \{i \in I : (\mathcal{M}_i, s_i) \vDash \varphi\} \in \mathcal{U}$$

In other words, for all $\varphi(\mathsf{x}_1, \mathsf{x}_2, \ldots, \mathsf{x}_k) \in Form_{\mathcal{L}}$ and all $g_1, g_2, \ldots, g_k \in \prod_{i \in I} M_i$, we have

$$(\mathcal{M}, [g_1], [g_2], \ldots, [g_k]) \vDash \varphi \text{ if and only if } \{i \in I : (\mathcal{M}_i, g_1(i), g_2(i), \ldots, g_k(i)) \vDash \varphi\} \in \mathcal{U}$$

In particular, for any $\sigma \in Sent_{\mathcal{L}}$, we have

$$\mathcal{M} \vDash \sigma \text{ if and only if } \{i \in I : \mathcal{M}_i \vDash \sigma\} \in \mathcal{U}.$$

Proof. The proof is by induction.

Suppose that $t_1, t_2 \in Term_{\mathcal{L}}$. Let $s_i \colon Var \to M_i$ be variable assignments. We then have

$$\begin{aligned}
(\mathcal{M}, \langle s_i \rangle_{i \in I}) \vDash = t_1 t_2 &\Leftrightarrow \overline{\langle s_i \rangle_{i \in I}}(t_1) = \overline{\langle s_i \rangle_{i \in I}}(t_2) \\
&\Leftrightarrow [i \mapsto \overline{s_i}(t_1)] = [i \mapsto \overline{s_i}(t_2)] \\
&\Leftrightarrow \{i \in I : \overline{s_i}(t_1) = \overline{s_i}(t_2)\} \in \mathcal{U} \\
&\Leftrightarrow \{i \in I : (\mathcal{M}_i, s_i) \vDash = t_1 t_2\} \in \mathcal{U}
\end{aligned}$$

Suppose that $\mathsf{R} \in \mathcal{R}_k$ and $t_1, t_2, \ldots, t_k \in Term_{\mathcal{L}}$. Let $s_i \colon Var \to M_i$ be variable assignments. We then have

$$\begin{aligned}
(\mathcal{M}, \langle s_i \rangle_{i \in I}) \vDash \mathsf{R} t_1 t_2 \cdots t_k &\Leftrightarrow (\overline{\langle s_i \rangle_{i \in I}}(t_1), \overline{\langle s_i \rangle_{i \in I}}(t_2), \ldots, \overline{\langle s_i \rangle_{i \in I}}(t_k)) \in \mathsf{R}^{\mathcal{M}} \\
&\Leftrightarrow ([i \mapsto \overline{s_i}(t_1)], [i \mapsto \overline{s_i}(t_2)], \ldots, [i \mapsto \overline{s_i}(t_k)]) \in \mathsf{R}^{\mathcal{M}} \\
&\Leftrightarrow \{i \in I : (\overline{s_i}(t_1), \overline{s_i}(t_2), \ldots, \overline{s_i}(t_k)) \in \mathsf{R}^{\mathcal{M}_i}\} \in \mathcal{U} \\
&\Leftrightarrow \{i \in I : (\mathcal{M}_i, s_i) \vDash \mathsf{R} t_1 t_2 \cdots t_k\} \in \mathcal{U}
\end{aligned}$$

Suppose that the result holds for φ and ψ. Let $s_i \colon Var \to M_i$ be variable assignments. We then have

$$\begin{aligned}
(\mathcal{M}, \langle s_i \rangle_{i \in I}) \vDash \varphi \wedge \psi &\Leftrightarrow (\mathcal{M}, \langle s_i \rangle_{i \in I}) \vDash \varphi \text{ and } (\mathcal{M}, \langle s_i \rangle_{i \in I}) \vDash \psi \\
&\Leftrightarrow \{i \in I : (\mathcal{M}_i, s_i) \vDash \varphi\} \in \mathcal{U} \text{ and } \{i \in I : (\mathcal{M}_i, s_i) \vDash \psi\} \in \mathcal{U} \\
&\Leftrightarrow \{i \in I : (\mathcal{M}_i, s_i) \vDash \varphi\} \cap \{i \in I : (\mathcal{M}_i, s_i) \vDash \psi\} \in \mathcal{U} \\
&\Leftrightarrow \{i \in I : (\mathcal{M}_i, s_i) \vDash \varphi \text{ and } (\mathcal{M}_i, s_i) \vDash \psi\} \in \mathcal{U} \\
&\Leftrightarrow \{i \in I : (\mathcal{M}_i, s_i) \vDash \varphi \wedge \psi\} \in \mathcal{U}
\end{aligned}$$

Suppose that the result holds for φ. Let $s_i \colon Var \to M_i$ be variable assignments. We then have

$$\begin{aligned}(\mathcal{M}, \langle s_i \rangle_{i \in I}) \vDash \neg\varphi &\Leftrightarrow (\mathcal{M}, \langle s_i \rangle_{i \in I}) \nvDash \varphi \\ &\Leftrightarrow \{i \in I : (\mathcal{M}_i, s_i) \vDash \varphi\} \notin \mathcal{U} \\ &\Leftrightarrow \{i \in I : (\mathcal{M}_i, s_i) \nvDash \varphi\} \in \mathcal{U} \\ &\Leftrightarrow \{i \in I : (\mathcal{M}_i, s_i) \vDash \neg\varphi\} \in \mathcal{U}\end{aligned}$$

Suppose that the result holds for φ. Let $s_i \colon Var \to M_i$ be variable assignments. We then have

$$\begin{aligned}(\mathcal{M}, \langle s_i \rangle_{i \in I}) \vDash \exists \mathsf{y}\varphi &\Leftrightarrow \text{There exists } a \in M \text{ such that } (\mathcal{M}, \langle s_i \rangle_{i \in I}[\mathsf{y} \Rightarrow a]) \vDash \varphi \\ &\Leftrightarrow \text{There exists } g \in \prod_{i \in I} M_i \text{ such that } (\mathcal{M}, \langle s_i \rangle_{i \in I}[\mathsf{y} \Rightarrow [g]]) \vDash \varphi \\ &\Leftrightarrow \text{There exists } g \in \prod_{i \in I} M_i \text{ such that } (\mathcal{M}, \langle s_i[\mathsf{y} \Rightarrow g(i)] \rangle_{i \in I}) \vDash \varphi \\ &\Leftrightarrow \text{There exists } g \in \prod_{i \in I} M_i \text{ such that } \{i \in I : (\mathcal{M}_i, s_i[\mathsf{y} \Rightarrow g(i)]) \vDash \varphi\} \in \mathcal{U} \\ &\Leftrightarrow \{i \in I : \text{ There exists } a \in M_i \text{ such that } (\mathcal{M}_i, s_i[\mathsf{y} \Rightarrow a]) \vDash \varphi\} \in \mathcal{U} \\ &\Leftrightarrow \{i \in I : (\mathcal{M}_i, s_i) \vDash \exists \mathsf{y}\varphi\} \in \mathcal{U}.\end{aligned}$$

□

Definition 11.3.15. *Let $\mathcal{N}$ be an $\mathcal{L}$-structure, let I be a set, and let $\mathcal{U}$ be an ultrafilter on I. If we let $\mathcal{M}_i = \mathcal{N}$ for all $i \in I$, then the ultraproduct $\mathcal{M} = \prod_{i \in I} \mathcal{M}_i / \mathcal{U}$ is called an* ultrapower *of $\mathcal{N}$.*

For example, let $\mathcal{L} = \{\mathsf{0}, \mathsf{1}, +, \cdot, <\}$, and let $\mathfrak{N} = (\mathbb{N}, 0, 1, +, \cdot, <)$. Let $I = \omega$, and let $\mathcal{U}$ be an ultrafilter on ω. Consider the corresponding ultrapower on $\mathcal{M}$. Elements of M are equivalence classes of infinite sequences of natural numbers. For example, $[(2, 2, 2, 2, \dots)]$, $[(0, 1, 0, 1, \dots)]$ and $[(0, 1, 2, 3, \dots)]$ are all elements of $\mathcal{M}$. Given any $\sigma \in Sent_{\mathcal{L}}$, we have $\mathcal{M} \vDash \sigma$ if and only if $\{i \in I : \mathcal{M}_i \vDash \sigma\} \in \mathcal{U}$. However, each $\mathcal{M}_i$ is just $\mathfrak{N}$, so since $\emptyset \notin \mathcal{U}$ and $\omega \in \mathcal{U}$, it follows that $\mathcal{M} \vDash \sigma$ if and only if $\mathfrak{N} \vDash \sigma$. Therefore, $\mathcal{M} \equiv \mathfrak{N}$, and so $\mathcal{M}$ is a model of $Th(\mathfrak{N})$.

Notice hat $\mathsf{0}^{\mathcal{M}} = [(0, 0, 0, 0, \dots)]$, that $\mathsf{1}^{\mathcal{M}} = [(1, 1, 1, 1, \dots)]$. Addition and multiplication on $\mathcal{M}$ are defined componentwise on the equivalence classes, and ordering is determined by taking two equivalence classes, and asking if the set of i where the i^{th} entry of the first element is less than the i^{th} entry of the second is an element of $\mathcal{U}$. However, this latter fact requires knowledge of $\mathcal{U}$. For example, let $a = [(0, 1, 0, 1, \dots)]$ and let $b = [(1, 0, 1, 0, \dots)]$. We know that $<^{\mathcal{M}}$ is a linear ordering on $\mathcal{M}$ (since $\mathcal{M}$ is a model of $Th(\mathfrak{N})$), so one of $a <^{\mathcal{M}} b$, $b <^{\mathcal{M}} a$, or $a =^{\mathcal{M}} b$ is true. The last is impossible because there is no i where the $a_i = b_i$ (where a_i is the i^{th} entry of $(0, 1, 0, 1, \dots)$, and similarly for b_i). Notice that

$$\{i \in \omega : a_i < b_i\} = \{2n : n \in \omega\} \quad \text{and} \quad \{i \in \omega : b_i < a_i\} = \{2n + 1 : n \in \omega\}$$

Now either $\{2n : n \in \omega\} \in \mathcal{U}$ or $\{2n + 1 : n \in \omega\} \in \mathcal{U}$, but not both, because $\mathcal{U}$ is an ultrafilter. In the former case, we have $a <^{\mathcal{M}} b$, while in the latter case we have $b <^{\mathcal{M}} a$.

We have a natural function $f \colon \mathbb{N} \to M$ given by letting $f(n) = [(n, n, n, n, \dots)]$ for all $n \in \mathbb{N}$. Notice that f is injective because if $m \neq n$, then the set of $i \in \omega$ where the i^{th} element of $(m, m, m, m, \dots)$ equals the i^{th} element of $(n, n, n, n, \dots)$ is the empty set, which is not in $\mathcal{U}$. Now if $\mathcal{U}$ is principal, then it is straightforward to check that f is also surjective, and in fact that it gives an isomorphism from $\mathfrak{N}$ to $\mathcal{M}$. Suppose instead that $\mathcal{U}$ is nonprincipal. Consider the element $[(0, 1, 2, 3, \dots)]$. For each $n \in \omega$, the set of places where $(0, 1, 2, 3, \dots)$ equals $(n, n, n, n, \dots)$ is a singleton, which is not in $\mathcal{U}$. Thus, $[(0, 1, 2, 3, \dots)] \notin \text{range}(f)$. In fact, since no finite set is in $\mathcal{U}$, we have $[(n, n, n, n, \dots)] < [(0, 1, 2, 3, \dots)]$ for each $n \in \omega$. Therefore, $\mathcal{M}$ is a nonstandard model of arithmetic, and $[(0, 1, 2, 3, \dots)]$ is an example of an infinite element.

We now use ultraproducts to give a another, purely semantic, proof of the Compactness Theorem for first-order logic. For simplicity of notation, we prove it for a set of sentences.

Theorem 11.3.16. *Let $\mathcal{L}$ be a language, and let $\Sigma \subseteq Sent_{\mathcal{L}}$. If every finite subset of Σ has a model, then Σ has a model.*

Proof. Let I be the set of all finite subsets of Σ. For each $\Psi \in I$, fix a model $\mathcal{M}_\Psi$ of Ψ. For each $\sigma \in \Sigma$, let $A_\sigma = \{\Psi \in I : \sigma \in \Psi\}$. Let $\mathcal{S} = \{A_\sigma : \sigma \in \Sigma\} \subseteq \mathcal{P}(I)$ and notice that $\mathcal{S}$ has the finite intersection property because

$$\{\sigma_1, \sigma_2, \ldots, \sigma_n\} \in A_{\sigma_1} \cap A_{\sigma_2} \cap \cdots \cap A_{\sigma_n}.$$

Since $\mathcal{S}$ has the finite intersection property, we can fix an ultrafilter $\mathcal{U}$ on I such that $\mathcal{S} \subseteq \mathcal{U}$. Let $\mathcal{M}$ be the corresponding ultraproduct $\mathcal{M} = \prod_{\Psi \in I} \mathcal{M}_\Psi / \mathcal{U}$. For any $\sigma \in \Sigma$, we then have that $A_\sigma \subseteq \{\Psi \in I : \mathcal{M}_\Psi \vDash \sigma\}$, hence $\{\Psi \in I : \mathcal{M}_\Psi \vDash \sigma\} \in \mathcal{U}$, and so $\mathcal{M} \vDash \sigma$. Therefore, $\mathcal{M}$ is a model of Σ. □

11.4 Exercises

1. (a) Let $\mathcal{F}$ be the set of all functions from $\mathbb{R}$ to $\mathbb{R}$. Show that $|\mathcal{F}| = 2^{2^{\aleph_0}}$.

   (b) Let $\mathcal{C}$ be the set of all continuous functions from $\mathbb{R}$ to $\mathbb{R}$. Show that $|\mathcal{C}| = 2^{\aleph_0}$.

   Together with the fact that $2^{\aleph_0} < 2^{2^{\aleph_0}}$, this gives the worst proof ever that there is a function $f : \mathbb{R} \to \mathbb{R}$ that is not continuous.

2. Suppose that V is a vector space over a field F. Let B_1 and B_2 be two bases of V over F, and suppose that at least one of B_1 or B_2 is infinite. Show that $|B_1| = |B_2|$. You may use the standard linear algebra fact that if B is a finite basis of V over F and $S \subseteq V$ is finite with $|S| > |B|$, then S is linearly dependent.

   Hint: Express each of the elements of B_2 as a finite linear combination of elements of B_1. How many total elements of B_1 are used in this way?

3. Let X be a set, and let $\mathcal{U}$ be an ultrafilter on X. Show that if $\mathcal{T} \subseteq \mathcal{P}(X)$ is finite, and $\bigcup \mathcal{T} \in \mathcal{U}$, then there exists $A \in \mathcal{T}$ such that $A \in \mathcal{U}$. Furthermore, show that if the elements of $\mathcal{T}$ are pairwise disjoint, then the A is unique.

4. Show that every ultrafilter on a finite set is principal.

5. We say that an ultrafilter $\mathcal{U}$ on a set X is σ-complete if $\bigcap_{n \in \omega} A_n \in \mathcal{U}$ whenever $A_n \in \mathcal{U}$ for all $n \in \omega$. Show that an ultrafilter on ω is σ-complete if and only if it is principal. (The question of whether there exists a nonprincipal σ-complete ultrafilter on any set is a very deep and interesting one. If such an ultrafilter exists on a set X, then X is ridiculously large.)

6. Let $L = \{\mathsf{P}\}$ where P is a unary relation symbol. For each $n \in \mathbb{N}^+$, let

$$\sigma_n = \exists \mathsf{x}_1 \exists \mathsf{x}_2 \ldots \exists \mathsf{x}_n \Big(\bigwedge_{1 \leq i < j \leq n} (\mathsf{x}_i \neq \mathsf{x}_j) \wedge \bigwedge_{1 \leq i \leq n} \mathsf{P}\mathsf{x}_i \Big)$$

$$\tau_n = \exists \mathsf{x}_1 \exists \mathsf{x}_2 \ldots \exists \mathsf{x}_n \Big(\bigwedge_{1 \leq i < j \leq n} (\mathsf{x}_i \neq \mathsf{x}_j) \wedge \bigwedge_{1 \leq i \leq n} \neg \mathsf{P}\mathsf{x}_i \Big)$$

   Let $\Sigma = \{\sigma_n : n \geq 1\} \cup \{\tau_n : n \geq 1\}$ (so Σ says that there infinitely many elements satisfying P and infinitely many not satisfying P), and let $T = Cn(\Sigma)$. Calculate $I(T, \aleph_\alpha)$ for each ordinal α.

   Hint: It may help to first think about $\alpha = 0$, $\alpha = 1$, and $\alpha = \omega$ to get the general pattern.

7. In the language $\mathcal{L} = \{\mathsf{0}, \mathsf{1}, +, \cdot, <\}$, let $\mathcal{N}$ be the $\mathcal{L}$-structure $(\mathbb{N}, 0, 1, +, \cdot, <)$. Show that $I(Th(\mathcal{N}), \aleph_0) = 2^{\aleph_0}$.

   Hint: Use Exercise 1 in Chapter 7.

8. Let $\mathcal{U}$ be a nonprincipal ultrafilter on ω. Suppose that $\langle a_n \rangle_{n \in \omega}$ is a bounded sequence of real numbers.

   (a) Show that there exists a unique real number ℓ, denoted by $\mathcal{U}\text{-}\lim a_n$, such that for all $\varepsilon > 0$, we have
   $$\{n \in \omega : |a_n - \ell| < \varepsilon\} \in \mathcal{U}.$$

   (b) Show that if $\lim a_n = \ell$, then $\mathcal{U}\text{-}\lim a_n = \ell$.

   It's not hard to show that $\mathcal{U}$-lim obeys the usual limit rules. Thus, a nonprincipal ultrafilter on ω gives a way to coherently define the limit of *any* bounded sequence of real numbers.

9. (**) Show that there is an uncountable subset of $\mathbb{R}$ which does not have a nonempty perfect subset.

Chapter 12

Defining Computable

One of the major triumphs of early 20^{th} century logic was the formulation of several (equivalent) precise definitions for what it means to say that a function $f\colon \mathbb{N} \to \mathbb{N}$ is computable. In our current age, many people have an intuitive sense of what computable means from experience with computer programming. However, it is challenging to turn that intuition into a concise formal mathematical treatment that is susceptible to a rigorous mathematical analysis. In this chapter, we will develop several historical attempts to define the computable functions. As we will see, our first approach will fail to capture all computable functions, but does describe an important subclass that includes the vast majority of the functions that arise in practice. We will then go on to properly define the computable functions after isolating the underlying issues with our first attempt.

12.1 Primitive Recursive Functions

Rather than restricting our attention to functions $f\colon \mathbb{N} \to \mathbb{N}$, it is easier to define the classes of functions that we are interested in by widening our scope to the collection of all functions $f\colon \mathbb{N}^n \to \mathbb{N}$ over all $n \in \mathbb{N}^+$.

Definition 12.1.1. *Let $\mathcal{F}$ be the set of all functions $f\colon \mathbb{N}^n \to \mathbb{N}$ for some $n \in \mathbb{N}^+$.*

Our first attempt to define the class of computable functions will follow the generation template that we have used many times. That is, we will start with some simple functions that are obviously computable, and then generate more complicated functions through the use of a couple of simple rules. We begin by describing the basic functions:

Definition 12.1.2. *We define the* initial functions *to be the following:*

1. *$\mathcal{O}\colon \mathbb{N} \to \mathbb{N}$ defined by $\mathcal{O}(x) = 0$.*
2. *$\mathcal{S}\colon \mathbb{N} \to \mathbb{N}$ defined by $\mathcal{S}(x) = x + 1$.*
3. *For each $i, n \in \mathbb{N}^+$ with $1 \leq i \leq n$, we define $\mathcal{I}_i^n\colon \mathbb{N}^n \to \mathbb{N}$ by $\mathcal{I}_i^n(x_1, x_2, \ldots, x_n) = x_i$.*

We denote the collection of initial functions by Init.

We now define a couple of ways to generate new elements of $\mathcal{F}$ from others. The first method is simply composition, but since we are are working with functions $f\colon \mathbb{N}^n \to \mathbb{N}$, we have to be a bit careful. For example, if we have functions $h\colon \mathbb{N}^2 \to \mathbb{N}$ and $g\colon \mathbb{N}^3 \to \mathbb{N}$, then we can not form $h \circ g$ because the domain of h does not match the codomain of g. However, if we have a function $h\colon \mathbb{N}^2 \to \mathbb{N}$ together with *two* functions $g_1, g_2\colon \mathbb{N}^3 \to \mathbb{N}$, then we can form the function $f(x, y, z) = h(g_1(x, y, z), g_2(x, y, z))$. In other words, in order

to form a composition, we allow several different functions on the inside, but we restrict to the case where they all have a common arity, and where the number of such inside functions matches the arity of the outer function.

Definition 12.1.3. *Let $m, n \in \mathbb{N}^+$, let $h\colon \mathbb{N}^m \to \mathbb{N}$, and let $g_1, g_2, \ldots, g_m\colon \mathbb{N}^n \to \mathbb{N}$. We define a new function* Compose$(h, g_1, g_2, \ldots, g_m)$ *as follows. We let $Compose(h, g_1, g_2, \ldots, g_m)$ be the function $f\colon \mathbb{N}^n \to \mathbb{N}$ defined by $f(\vec{x}) = h(g_1(\vec{x}), g_2(\vec{x}), \ldots, g_m(\vec{x}))$ for all $\vec{x} \in \mathbb{N}^n$.*

Notice that *Compose* is not actually a function from some $\mathcal{F}^k$ to $\mathcal{F}$. For each fixed $m \in \mathbb{N}^+$, we *almost* have a function $Compose_m$ from $\mathcal{F}^{m+1}$ to $\mathcal{F}$, but it is only defined in the case where the arity of the first function is m, and the latter m inputs all have the same arity. In all other cases, we can either leave $Compose_m$ undefined (and thus use the more general definition of a generating function), or we can simply output the initial function $\mathcal{O}$ when the inputs are not appropriate. Regardless, simply note that *Compose* is technically comprised of infinitely many generating functions.

The other method we use to generate new elements of $\mathcal{F}$ from old ones is step recursion. We first explored how to define functions recursively on free generating systems in Theorem 2.4.5. In the simpler context of ω, we formalized this idea in set theory with Theorem 9.5.1, and then generalized it to allow parameters in Theorem 9.5.2. Taking this latter description, but allowing multiple natural number parameters (rather than coding them as one n-tuple p in Theorem 9.5.2) leads to the following.

Definition 12.1.4. *Let $n \in \mathbb{N}^+$, let $g\colon \mathbb{N}^n \to \mathbb{N}$, and let $h\colon \mathbb{N}^{n+2} \to \mathbb{N}$. We let* PrimRec$(g, h)$ *be the unique function $f\colon \mathbb{N}^{n+1} \to \mathbb{N}$ defined by letting*

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

for all $\vec{x} \in \mathbb{N}^n$ and all $y \in \mathbb{N}$.

Notice that *PrimRec* is almost a function from $\mathcal{F}^2$ to $\mathcal{F}$, but that $PrimRec(g, h)$ is only defined when the arity of the second argument is two more than the arity of the first argument. Again, we can either use the more general definition of a generating system, or simply output $\mathcal{O}$ in situations when the inputs do not satisfy this assumption.

Now that we have defined the initial functions and two ways (or technically infinitely many if we properly break up *Compose* into the infinitely many $Compose_m$ functions), we can now define an interesting class of functions through generation.

Definition 12.1.5. *The collection of* primitive recursive functions *is the subset of $\mathcal{F}$ generated by starting with Init, and generating using Compose and PrimRec.*

At first, it is difficult to imagine what the collection of primitive recursive functions looks like. On the one hand, we start with only very simple functions and just close off under certain relatively tame operations. On the other hand, we've seen that many functions on ω are defined though repeated applications of Theorem 9.5.2. We will soon see that the collection of primitive recursive functions is quite robust and contains the vast majority of the elements of $\mathcal{F}$ that arise naturally. In order to show this, we will prove in the coming sections that the primitive recursive functions are closed under many other important operations.

Before we start proving theorems about the class of primitive recursive functions, we first pause to notice that every primitive recursive function is "intuitively computable". To see this, we begin by noting that the initial functions are all trivially "intuitively computable". Next, notice that if we have a way to compute a function $h\colon \mathbb{N}^m \to \mathbb{N}$, and also ways to compute functions $g_1, g_2, \ldots, g_m\colon \mathbb{N}^n \to \mathbb{N}$, then we have a straightforward way to compute the function $f = Compose(h, g_1, g_2, \ldots, g_m)$. Namely, on input $\vec{x}$, to compute $f(\vec{x})$, first compute $g_1(\vec{x}), g_2(\vec{x}), \ldots, g_m(\vec{x})$, and then take these m results and feed them as input into the method to compute h. In other words, the collection of "intuitively computable" functions is closed under *Compose*.

What about $PrimRec$? Suppose that we have a mechanism to compute both a function $g\colon \mathbb{N}^n \to \mathbb{N}$ and a function $h\colon \mathbb{N}^{n+2} \to \mathbb{N}$. We claim that we can compute the function $f = PrimRec(g, h)$ by using a simple loop. For example, if we know how to compute a function $g\colon \mathbb{N} \to \mathbb{N}$ and know how to compute a function $h\colon \mathbb{N}^3 \to \mathbb{N}$, then here is a mechanism to compute $f = PrimRec(g, h)\colon \mathbb{N}^2 \to \mathbb{N}$. To determine $f(6, 2)$, we first compute the value $f(6, 0) = g(6)$ by using the mechanism for g. We then compute $f(6, 1) = h(6, 0, f(6, 0))$ by using our computed value of $f(6, 0)$ together with the mechanism for computing h. Finally, we compute $f(6, 2) = h(6, 1, f(6, 1))$ by using our computed value of $f(6, 1)$ together with the mechanism for computing h. In general, to compute $f(x, y)$, we simply start by computing $g(x)$, and then loop through applying h a total of y many times. By generalizing this argument to any arity, it follows that the collection of "intuitively computable" functions is closed under $PrimRec$.

We now turn to the task of showing that class of primitive recursive functions includes many natural "intuitively computable" functions. We begin by showing that some very simple functions are primitive recursive.

Proposition 12.1.6. *For every $k \in \mathbb{N}$ and every $n \in \mathbb{N}^+$, the constant function $\mathcal{C}_k^n\colon \mathbb{N}^n \to \mathbb{N}$ given by $\mathcal{C}_k^n(\vec{x}) = k$ for all $\vec{x} \in \mathbb{N}^n$ is primitive recursive.*

Proof. We first handle the case when $n = 1$ by using induction on k. For the base case, notice that $\mathcal{C}_0^1 = \mathcal{O}$, so $\mathcal{C}_0^1$ is primitive recursive. For the inductive step, assume that $\mathcal{C}_k^1$ is primitive recursive. Since $\mathcal{C}_{k+1}^1 = Compose(\mathcal{S}, \mathcal{C}_k^1)$, it follows that $\mathcal{C}_{k+1}^1$ is primitive recursive. Therefore, $\mathcal{C}_k^1$ is primitive recursive for all $k \in \mathbb{N}$.

We now turn to the general case. Given any $k \in \mathbb{N}$ and $n \in \mathbb{N}^+$, we have that $\mathcal{C}_k^n = Compose(\mathcal{C}_k^1, \mathcal{I}_1^n)$, so $\mathcal{C}_k^n$ is primitive recursive. □

Proposition 12.1.7. *Each of the following functions are primitive recursive:*

1. *The addition function $f\colon \mathbb{N}^2 \to \mathbb{N}$ given by $f(x, y) = x + y$.*
2. *The multiplication function $f\colon \mathbb{N}^2 \to \mathbb{N}$ given by $f(x, y) = x \cdot y$.*
3. *The exponentiation function $f\colon \mathbb{N}^2 \to \mathbb{N}$ given by $f(x, y) = x^y$.*

Proof.

1. Notice that
$$\begin{aligned} f(x, 0) &= x \\ f(x, y + 1) &= f(x, y) + 1 \end{aligned}$$
for all $x, y \in \mathbb{N}$. Thus, defining $g\colon \mathbb{N} \to \mathbb{N}$ by $g(x) = x$, and defining $h\colon \mathbb{N}^3 \to \mathbb{N}$ by $h(x, y, a) = a + 1$, we have $f = PrimRec(g, h)$. Now $g = \mathcal{I}_1^1$, so g is primitive recursive. Also, since $h = Compose(\mathcal{S}, \mathcal{I}_3^3)$, we have that h is primitive recursive. It follows that $f = PrimRec(g, h)$ is primitive recursive.

2. Notice that
$$\begin{aligned} f(x, 0) &= 0 \\ f(x, y + 1) &= f(x, y) + x \end{aligned}$$
for all $x, y \in \mathbb{N}$. Thus, defining $g\colon \mathbb{N} \to \mathbb{N}$ by $g(x) = 0$, and defining $h\colon \mathbb{N}^3 \to \mathbb{N}$ by $h(x, y, a) = a + x$, we have $f = PrimRec(g, h)$. Now $g = \mathcal{O}$, so g is primitive recursive. Also, since $h = Compose(+, \mathcal{I}_3^3, \mathcal{I}_1^3)$, we have that h is primitive recursive by part 1. It follows that $f = PrimRec(g, h)$ is primitive recursive.

3. Notice that

$$f(x,0) = 1$$
$$f(x,y+1) = f(x,y) \cdot x$$

for all $x, y \in \mathbb{N}$. Thus, defining $g\colon \mathbb{N} \to \mathbb{N}$ by $g(x) = 1$, and defining $h\colon \mathbb{N}^3 \to \mathbb{N}$ by $h(x,y,a) = a \cdot x$, we have $f = PrimRec(g,h)$. Now $g = \mathcal{C}_1^1$, so g is primitive recursive by Proposition 12.1.6. Also, since $h = Compose(\cdot, \mathcal{I}_3^3, \mathcal{I}_1^3)$, we have that h is primitive recursive by part 2. It follows that $f = PrimRec(g,h)$ is primitive recursive.

□

In the above proof that addition is primitive recursive, we could have simply started with the function

$$PrimRec(\mathcal{I}_1^1, Compose(\mathcal{S}, \mathcal{I}_3^3))$$

up front, and then verified that this function was indeed $+$. However, it is usually easier to follow an argument where we start with the recursive properties of f, and then notice that the underlying g and h are primitive recursive.

As we work with more complicated functions, we will not typically write out full witnessing sequences demonstrating that they are primitive recursive. Nonetheless, there is still some value in noticing that the addition function is explicitly given by

$$PrimRec(\mathcal{I}_1^1, Compose(\mathcal{S}, \mathcal{I}_3^3)).$$

For example, from this perspective, we can go back to the above informal description of how primitive recursion corresponds to a simple loop, and notice that we can compute $x + y$ by starting with x, and looping through application of the function $\mathcal{S}$ a total of y many times.

Before moving on, we pause to note that in the definition of $Compose$, the restriction that the inside functions $g_1, g_2, \ldots, g_m$ have the same arity and identical ordering of input variables is simply for convenience. To illustrate, suppose that we have primitive recursive functions $g_1\colon \mathbb{N} \to \mathbb{N}$, $g_2\colon \mathbb{N}^2 \to \mathbb{N}$, and $h\colon \mathbb{N}^2 \to \mathbb{N}$. Consider a situation where we want to define a function $f\colon \mathbb{N}^3 \to \mathbb{N}$ by letting

$$f(x,y,z) = h(g_1(y), g_2(z,x)).$$

Although this is a type of composition, notice the inside functions do not have the same arities, do not even use the same inputs, and also change the ordering of the input variables of f. However, f is still primitive recursive. To see this, define $k_1\colon \mathbb{N}^3 \to \mathbb{N}$ by letting $k_1(x,y,z) = g_1(y)$, and notice that $k_1(x,y,z) = g_1(\mathcal{I}_2^3(x,y,z))$. Thus, $k_1 = Compose(g_1, \mathcal{I}_2^3)$, and hence k_1 is primitive recursive. Also, define $k_2\colon \mathbb{N}^3 \to \mathbb{N}$ by letting $k_2(x,y,z) = g_2(z,x)$, and notice that $k_2(x,y,z) = g_2(\mathcal{I}_3^3(x,y,z), \mathcal{I}_1^3(x,y,z))$. Thus, $k_2 = Compose(g_2, \mathcal{I}_3^3, \mathcal{I}_1^3)$, and hence k_2 is primitive recursive. Finally, since

$$\begin{aligned} f(x,y,z) &= h(g_1(y), g_2(z,x)) \\ &= h(k_1(x,y,z), k_2(x,y,z)), \end{aligned}$$

we see that $f = Compose(h, k_1, k_2)$, so f is primitive recursive.

We next handle another small annoyance. Using our definition of $PrimRec$ directly, we have no way to define a recursive function $f\colon \mathbb{N} \to \mathbb{N}$ by setting $f(0)$ equal to some number and then defining $f(y+1)$ in terms of a known primitive recursive function of y and $f(y)$. The small obstacle is that all of our objects are functions, so we required the g in $PrimRec$ to be a function, not a number. However, we can easily prove that such definitions give primitive recursive functions by adding on a "dummy" variable and projecting down at the end.

Proposition 12.1.8. *Let $a \in \mathbb{N}$ and let $h\colon \mathbb{N}^2 \to \mathbb{N}$ be primitive recursive. Let $f\colon \mathbb{N} \to \mathbb{N}$ be defined by*

$$\begin{aligned} f(0) &= a \\ f(y+1) &= h(y, f(y)) \end{aligned}$$

for all $y \in \mathbb{N}$. Then f is primitive recursive.

The idea is first to go up to a function of two variables $f'\colon \mathbb{N}^2 \to \mathbb{N}$ defined by

$$\begin{aligned} f'(x,0) &= a \\ f'(x,y+1) &= h'(x,y,f'(x,y)), \end{aligned}$$

where $h'\colon \mathbb{N}^3 \to \mathbb{N}$ is given by $h'(x,y,a) = h(y,a)$. This is something we can handle, after which we can strip off the extraneous first variable to get that $f(y) = f'(0,y)$ is primitive recursive.

Proof. Let $g' = \mathcal{C}_a^1\colon \mathbb{N} \to \mathbb{N}$, and note that g' is primitive recursive by Proposition 12.1.6. Define $h'\colon \mathbb{N}^3 \to \mathbb{N}$ by letting $h'(x,y,a) = h(y,a)$, and notice that $h' = Compose(h, \mathcal{I}_2^3, \mathcal{I}_3^3)$, so h' is also primitive recursive. Letting $f' = PrimRec(g', h')$, we then have that f' is primitive recursive.

We now argue that $f(y) = f'(0,y)$ for all $y \in \mathbb{N}$ by induction. For the base case, notice that

$$\begin{aligned} f(0) &= a \\ &= g'(0) \\ &= f'(0,0). \end{aligned}$$

Assuming that $f(y) = f'(0,y)$, we then have

$$\begin{aligned} f(y+1) &= h(y, f(y)) \\ &= h(y, f'(0,y)) \\ &= h'(0,y,f'(0,y)) \\ &= f'(0,y+1). \end{aligned}$$

Therefore, we have $f(y) = f'(0,y)$ for all $y \in \mathbb{N}$. It follows that $f = Compose(f', \mathcal{O}, \mathcal{I}_1^1)$, and hence f is primitive recursive. □

With this result in hand, we can now argue that several other fundamental functions are primitive recursive.

Proposition 12.1.9. *The factorial function $f\colon \mathbb{N} \to \mathbb{N}$ is primitive recursive.*

Proof. Notice that

$$\begin{aligned} f(0) &= 0 \\ f(y+1) &= (y+1) \cdot f(y) \end{aligned}$$

for all $y \in \mathbb{N}$. Define $h\colon \mathbb{N}^2 \to \mathbb{N}$ by $h(y,a) = (y+1) \cdot a$. Since $h = Compose(\cdot, Compose(\mathcal{S}, \mathcal{I}_1^2), \mathcal{I}_2^2)$, we have that h is primitive recursive by Proposition 12.1.7. Therefore, f is primitive recursive by Proposition 12.1.8. □

Although the function $f(x,y) = x - y$ is not a function from $\mathbb{N}^2$ to $\mathbb{N}$, we now argue that the "truncated minus" function is primitive recursive.

Proposition 12.1.10. *Each of the following functions are primitive recursive:*

1. *The predecessor function* $Pred\colon \mathbb{N} \to \mathbb{N}$ *defined by* $Pred(0) = 0$ *and* $Pred(n) = n - 1$ *for all* $n > 0$.

2. *The function* $f\colon \mathbb{N}^2 \to \mathbb{N}$ *defined by*

$$f(x,y) = \begin{cases} x - y & \text{if } x \geq y \\ 0 & \text{otherwise.} \end{cases}$$

Proof. 1. We have

$$\begin{aligned} Pred(0) &= 0 \\ Pred(y+1) &= y \end{aligned}$$

for all $y \in \mathbb{N}$. Define $h\colon \mathbb{N}^2 \to \mathbb{N}$ by $h(y,a) = y$, and notice that $h = \mathcal{I}_1^2$ is primitive recursive. Therefore, $Pred$ is primitive recursive by Proposition 12.1.8.

2. Notice that

$$\begin{aligned} f(x,0) &= x \\ f(x,y+1) &= Pred(f(x,y)) \end{aligned}$$

for all $x, y \in \mathbb{N}$. Thus, defining $g\colon \mathbb{N} \to \mathbb{N}$ by $g(x) = x$, and defining $h\colon \mathbb{N}^3 \to \mathbb{N}$ by $h(x,y,a) = Pred(a)$, we have $f = PrimRec(g,h)$. Now $g = \mathcal{I}_1^1$, so g is primitive recursive. Also, since $h = Compose(Pred, \mathcal{I}_3^3)$, we have that h is primitive recursive by Proposition 12.1.10. It follows that $f = PrimRec(g,h)$ is primitive recursive.

□

Definition 12.1.11. *We use the notation* $x \dot{-} y$ *for the function* $f(x,y)$ *defined in part 2 of Proposition 12.1.10, i.e.*

$$x \dot{-} y = \begin{cases} x - y & \text{if } x \geq y \\ 0 & \text{otherwise.} \end{cases}$$

The function $\dot{-}$ *is sometimes called the* monus *function.*

Proposition 12.1.12. *The following functions are primitive recursive:*

1. $IsZero\colon \mathbb{N} \to \mathbb{N}$ *defined by*

$$IsZero(x) = \begin{cases} 1 & \text{if } x = 0 \\ 0 & \text{otherwise.} \end{cases}$$

2. $IsNonzero\colon \mathbb{N} \to \mathbb{N}$ *defined by*

$$IsNonzero(x) = \begin{cases} 1 & \text{if } x \neq 0 \\ 0 & \text{otherwise.} \end{cases}$$

Proof. Notice that $IsZero(x) = 1 \dot{-} x$ for every $x \in \mathbb{N}$, so $IsZero = Compose(\dot{-}, \mathcal{C}_1^1, \mathcal{I}_1^1)$, and hence $IsZero$ is primitive recursive by Proposition 12.1.6 and Proposition 12.1.10. Next notice that $IsNonzero = Compose(IsZero, IsZero)$, so $IsNonzero$ is primitive recursive as well. □

Proposition 12.1.13. *The following functions are primitive recursive:*

1. *The function* $Gt\colon \mathbb{N}^2 \to \mathbb{N}$ *defined by*

$$Gt(x,y) = \begin{cases} 1 & \text{if } x > y \\ 0 & \text{otherwise.} \end{cases}$$

2. *The function* $Lt\colon \mathbb{N}^2 \to \mathbb{N}$ *defined by*

$$Lt(x,y) = \begin{cases} 1 & \text{if } x < y \\ 0 & \text{otherwise.} \end{cases}$$

3. *The function* $Eq\colon \mathbb{N}^2 \to \mathbb{N}$ *defined by*

$$Eq(x,y) = \begin{cases} 1 & \text{if } x = y \\ 0 & \text{otherwise.} \end{cases}$$

4. *The function* $Neq\colon \mathbb{N}^2 \to \mathbb{N}$ *defined by*

$$Neq(x,y) = \begin{cases} 1 & \text{if } x \neq y \\ 0 & \text{otherwise.} \end{cases}$$

Proof. Notice that for all $x, y \in \mathbb{N}$, we have that $x > y$ if and only if $x \dot{-} y \neq 0$, which is if and only if $IsNonzero(x \dot{-} y) = 1$. Therefore, $Gt = Compose(IsNonzero, \dot{-})$, and so Gt is primitive recursive by Proposition 12.1.12 and Proposition 12.1.10. Next notice that $Lt(x,y) = Gt(y,x)$, so $Lt = Compose(Gt, \mathcal{I}_2^2, \mathcal{I}_1^2)$, and hence Lt is primitive recursive.

Moving on to Eq, we have that $x = y$ if and only if both $x \not> y$ and $x \not< y$. Thus,

$$Eq(x,y) = \begin{cases} 1 & \text{if } Gt(x,y) = 0 \text{ and } Lt(x,y) = 0 \\ 0 & \text{otherwise,} \end{cases}$$

and so

$$Eq(x,y) = \begin{cases} 1 & \text{if } Gt(x,y) + Lt(x,y) = 0 \\ 0 & \text{otherwise.} \end{cases}$$

Therefore, $Eq = Compose(IsZero, Compose(+, Gt, Lt))$, so Eq is primitive recursive by Proposition 12.1.12.

Finally, notice that $Neq(x,y) = IsZero(Eq(x,y))$ for all $x, y \in \mathbb{N}$, so $Neq = Compose(IsZero, Eq)$, and hence Neq is primitive recursive by Proposition 12.1.12. □

We are building a theory where our basic objects are functions $f\colon \mathbb{N}^n \to \mathbb{N}$, but we can also use our theory of functions to talk about subsets of $\mathbb{N}^n$ by working with their characteristic functions.

Definition 12.1.14. *Let $R \subseteq \mathbb{N}^n$. We say that R is primitive recursive if its characteristic function, i.e. the function $\chi_R\colon \mathbb{N}^n \to \mathbb{N}$ given by*

$$\chi_R(\vec{x}) = \begin{cases} 1 & \text{if } \vec{x} \in R \\ 0 & \text{otherwise,} \end{cases}$$

is primitive recursive.

For example, we can rephrase the statement that $IsZero$ is a primitive recursive function by instead saying that $\{0\} \subseteq \mathbb{N}$ is a primitive recursive relation. Also, we can look back on Proposition 12.1.13 as saying that the equality and various inequality relations on $\mathbb{N}^2$ are primitive recursive. In other words, by adopting this small change in language and perspective when talking about $\{0,1\}$-valued functions, we can often state results more succinctly. More importantly, we can state theorems about $\{0,1\}$-valued functions in set-theoretic language that would be more cumbersome in more function-theoretic language, such as in the following fundamental result.

Proposition 12.1.15. *Let $R, S \subseteq \mathbb{N}^n$ be primitive recursive relations.*

1. *The relation $\mathbb{N}^n \backslash R$ is primitive recursive.*
2. *The relation $R \cap S$ is primitive recursive.*
3. *The relation $R \cup S$ is primitive recursive.*

Proof. Since we have $\chi_{\mathbb{N}^n \backslash R}(\vec{x}) = 1$ if and only if $\chi_R(\vec{x}) = 0$, it follows that $\chi_{\mathbb{N}^n \backslash R} = Compose(IsZero, \chi_R)$, so $\mathbb{N}^n \backslash R$ is primitive recursive.

Next notice $\chi_{R \cap S}(\vec{x}) = \chi_R(\vec{x}) \cdot \chi_S(\vec{x})$, so $\chi_{R \cap S} = Compose(\cdot, \chi_R, \chi_S)$, and hence $R \cap S$ is primitive recursive.

Finally, we have $\chi_{R \cup S}(\vec{x}) > 0$ if and only if either $\chi_R(\vec{x}) > 0$ or $\chi_S(\vec{x}) > 0$, which is if and only if $\chi_R(\vec{x}) + \chi_S(\vec{x}) > 0$. Therefore, $\chi_{R \cup S} = Compose(IsNonzero, Compose(+, \chi_R, \chi_S))$, and hence $R \cup S$ is primitive recursive. □

Proposition 12.1.16.

1. *For each $k \in \mathbb{N}$, the relation $R = \{k\}$ is primitive recursive.*
2. *For each $\vec{k} \in \mathbb{N}^n$, the relation $R = \{\vec{k}\}$ is primitive recursive.*
3. *Every finite or cofinite relation (of some $\mathbb{N}^n$) is primitive recursive.*

Proof. For each $k \in \mathbb{N}$, we have that $\chi_{\{k\}} = Compose(Eq, \mathcal{I}_1^1, \mathcal{C}_k^1)$, so $\{k\}$ is primitive recursive by Proposition 12.1.6 and Proposition 12.1.13.

Let $\vec{k} \in \mathbb{N}^n$ be arbitrary, say $\vec{k} = (k_1, k_2, \ldots, k_n)$ where each $k_i \in \mathbb{N}$. By part 1, the functions $\chi_{\{k_i\}}$ are each primitive recursive. Now $\chi_{\{\vec{k}\}}(\vec{x}) = 1$ if and only if $\chi_{\{k_i\}}(\mathcal{I}_i^n(\vec{x})) = 1$ for all i, which is if and only if $(\chi_{\{k_i\}} \circ \mathcal{I}_i^n)(\vec{x}) = 1$ for all i. Therefore, $\chi_{\{\vec{k}\}} = (\chi_{\{k_1\}} \circ \mathcal{I}_1^n) \cdot (\chi_{\{k_2\}} \circ \mathcal{I}_2^n) \cdots (\chi_{\{k_n\}} \circ \mathcal{I}_n^n)$. Since each $\chi_{\{k_i\}} \circ \mathcal{I}_i^n = Compose(\chi_{\{k_i\}}, \mathcal{I}_i^n)$ is primitive recursive and $\cdot$ is primitive recursive, it follows that $\chi_{\{\vec{k}\}}$ is a primitive recursive function, and hence $\{\vec{k}\}$ is a primitive recursive relation.

We now prove 3. For finite relations, this follows from part 2 together with the fact that primitive recursive relations are closed under finite unions by Proposition 12.1.15. Since the primitive recursive relations are closed under complement by Proposition 12.1.15, we conclude that cofinite relations are also primitive recursive. □

Before moving on, we establish one implication between when a function is a primitive recursive and when its graph is a primitive recursive relation. It turns out that the converse to this statement is false, and this failure is closely connected to the reason why the primitive recursive functions do not exhaust the collection of all "intuitively computable" functions. We will return to that issue later.

Proposition 12.1.17. *If $f : \mathbb{N}^n \to \mathbb{N}$ is a primitive recursive function, then $graph(f) \subseteq \mathbb{N}^{n+1}$ is a primitive recursive relation.*

Proof. Notice that

$$\chi_{graph(f)}(\vec{x}, y) = \begin{cases} 1 & \text{if } f(\vec{x}) = y \\ 0 & \text{otherwise.} \end{cases}$$

It follows that $\chi_{graph(f)} = Compose(Eq, Compose(f, \mathcal{I}_1^{n+1}, \mathcal{I}_2^{n+1}, \cdots, \mathcal{I}_n^{n+1}), \mathcal{I}_{n+1}^{n+1})$, so $graph(f)$ is a primitive recursive relation by Proposition 12.1.13. □

Up until this point, we've spent a great deal of time to show that some of the simplest functions and relations imaginable are indeed primitive recursive. However, we are now ready to prove that the primitive recursive functions and relations are closed under more complicated operations, which will significantly streamline later arguments. We start with a couple of simple results about bounded sums and products.

Proposition 12.1.18. *Let $n \in \mathbb{N}$, and let $f \colon \mathbb{N}^{n+1} \to \mathbb{N}$ be primitive recursive.*

1. *The function $f' \colon \mathbb{N}^{n+1} \to \mathbb{N}$ given by $f'(\vec{x}, y) = \sum_{z<y} f(\vec{x}, z)$ is primitive recursive.*
2. *The function $f' \colon \mathbb{N}^{n+1} \to \mathbb{N}$ given by $f'(\vec{x}, y) = \prod_{z<y} f(\vec{x}, z)$ is primitive recursive.*

Proof. We prove 1 (the proof of 2 is similar). We first handle the case where $n > 0$. Notice that

$$\begin{aligned} f'(\vec{x}, 0) &= 0 \\ f'(\vec{x}, y+1) &= f'(\vec{x}, y) + f(\vec{x}, y) \end{aligned}$$

for all $\vec{x} \in \mathbb{N}^n$ and all $y \in \mathbb{N}$. Thus, defining $g \colon \mathbb{N}^n \to \mathbb{N}$ by $g(\vec{x}) = 0$, and defining $h \colon \mathbb{N}^{n+2} \to \mathbb{N}$ by $h(\vec{x}, y, a) = a + f(\vec{x}, y)$, we have $f' = PrimRec(g, h)$. Now $g = Compose(\mathcal{O}, \mathcal{I}_1^n)$, so g is primitive recursive. Also, since $h = Compose(+, \mathcal{I}_{n+2}^{n+2}, Compose(f, \mathcal{I}_1^{n+2}, \mathcal{I}_2^{n+2}, \ldots, \mathcal{I}_{n+1}^{n+2}))$, we have that h is primitive recursive. It follows that $f' = PrimRec(g, h)$ is primitive recursive.

For the case where $n = 0$, notice that

$$\begin{aligned} f'(0) &= 0 \\ f'(y+1) &= f'(y) + f(y) \end{aligned}$$

for all $y \in \mathbb{N}$. Define $h \colon \mathbb{N}^2 \to \mathbb{N}$ by $h(y, a) = a + y$, and notice that $h = Compose(+, \mathcal{I}_2^2, \mathcal{I}_1^2)$ is primitive recursive. Therefore, f' is primitive recursive by Proposition 12.1.8. □

Using *Compose*, we can extend this result to bounded sums and products where the bound is some primitive recursive function of the other variables.

Corollary 12.1.19. *Let $n \in \mathbb{N}$, and let $f, g \colon \mathbb{N}^{n+1} \to \mathbb{N}$ be primitive recursive.*

1. *The function $f' \colon \mathbb{N}^{n+1} \to \mathbb{N}$ given by $f'(\vec{x}, y) = \sum_{z<g(\vec{x},y)} f(\vec{x}, z)$ is primitive recursive.*
2. *The function $f' \colon \mathbb{N}^{n+1} \to \mathbb{N}$ given by $f'(\vec{x}, y) = \prod_{z<g(\vec{x},y)} f(\vec{x}, z)$ is primitive recursive.*

Proof. We prove 1 (the proof of 2 is similar). Define $h \colon \mathbb{N}^{n+1} \to \mathbb{N}$ by $h(\vec{x}, y) = \sum_{z<y} f(\vec{x}, z)$, and note that h is primitive recursive by Proposition 12.1.18. Since $f'(\vec{x}, y) = h(\vec{x}, g(\vec{x}, y))$ for all $\vec{x} \in \mathbb{N}^n$ and all $y \in \mathbb{N}$, we have

$$f' = Compose(h, \mathcal{I}_1^{n+1}, \mathcal{I}_2^{n+1}, \ldots, \mathcal{I}_n^{n+1}, g),$$

so f' is primitive recursive. □

We are finally ready to prove one of the fundamental closure properties of primitive recursive relations. Intuitively, it says that whenever we perform a primitive recursively bounded search on a primitive recursive relation, the result is still a primitive recursive relation.

Proposition 12.1.20. *Let $n \in \mathbb{N}$, let $g\colon \mathbb{N}^{n+1} \to \mathbb{N}$ be a primitive recursive function, and let $R \subseteq \mathbb{N}^{n+1}$ be a primitive recursive relation.*

1. *The relation $S \subseteq \mathbb{N}^{n+1}$ defined by*

$$S = \{(\vec{x}, y) \in \mathbb{N}^{n+1} : (\exists z < g(\vec{x}, y))R(\vec{x}, z)\}$$

   is primitive recursive.

2. *The relation $S \subseteq \mathbb{N}^{n+1}$ defined by*

$$S = \{(\vec{x}, y) \in \mathbb{N}^{n+1} : (\forall z < g(\vec{x}, y))R(\vec{x}, z)\}$$

   is primitive recursive.

Proof. For part 1, we have

$$\chi_S(\vec{x}, y) = IsNonzero\left(\sum_{z<g(\vec{x},y)} \chi_R(\vec{x}, z))\right),$$

so χ_S is primitive recursive by Proposition 12.1.12 and Corollary 12.1.19.

For part 2, notice that

$$\chi_S(\vec{x}, y) = IsNonzero\left(\prod_{z<g(\vec{x},y)} \chi_R(\vec{x}, z))\right),$$

so χ_S is similarly primitive recursive. □

We have omitted some routine detail in the above proof. More formally, we can argue the proof of part 1 as follows. Define $h\colon \mathbb{N}^{n+1} \to \mathbb{N}$ by

$$h(\vec{x}, y) = \sum_{z<g(\vec{x},y)} \chi_R(\vec{x}, z),$$

and notice that h is primitive recursive by Corollary 12.1.19. Since $\chi_S = Compose(IsNonzero, h)$, it follows that χ_S is primitive recursive. However, we will continue to omit such straightforward formalities in the future.

Proposition 12.1.21. *The set $Divides = \{(x, y) \in \mathbb{N}^2 : x \text{ divides } y\}$ is primitive recursive.*

Proof. Notice that

$$(x, y) \in Divides \Leftrightarrow (\exists w < y + 1)[x \cdot w = y].$$

Since *Divides* is obtained from a primitive recursive relation by an existential quantification with a primitive recursive bound, it follows from Proposition 12.1.20 that *Divides* is primitive recursive. □

As above, we have omitted a few details in the above argument. For example, we simply stated the relation $R = \{(x, y, w) \in \mathbb{N}^3 : x \cdot w = y\}$ is primitive recursive. There are at least two ways to work this out carefully:

- One way to argue this is to notice that R is, up to permutation of the variables, the graph of the primitive recursive multiplication function, and so we can appeal to Proposition 12.1.17. More formally, the actual graph of the multiplication function $G = \{(x, w, y) \in \mathbb{N}^3 : x \cdot w = y\}$ is primitive recursive by Proposition 12.1.17, and since $\chi_R = Compose(\chi_G, \mathcal{I}_1^3, \mathcal{I}_3^3, \mathcal{I}_2^3)$, it follows that R is primitive recursive. In general, as alluded to in the discussion after the proof of Proposition 12.1.7, we can use $Compose$ and the $\mathcal{I}_i^n$ functions to argue that if f is primitive recursive, and g is obtained from f by permuting the variables, then g is also primitive recursive.
- Alternatively, instead of appealing to Proposition 12.1.17, we can directly use the fact that $\chi_R = Compose(Eq, Compose(\cdot, \mathcal{I}_1^3, \mathcal{I}_3^3), \mathcal{I}_2^3)$, so R is primitive recursive.

Now that we have carefully established that R is primitive recursive, we notice that the bound on the quantifier is given by the function $g(x, y) = y + 1$. Since $g = Compose(\mathcal{S}, \mathcal{I}_2^2)$ is primitive recursive, we can now appeal to Proposition 12.1.20 to conclude that D is indeed primitive recursive.

Proposition 12.1.22. *The set $Prime \subseteq \mathbb{N}$ of prime numbers is primitive recursive.*

Proof. Notice that

$$n \in Prime \Leftrightarrow (n > 1) \wedge \neg(\exists k < n)(\exists m < n)[k \cdot m = n].$$

Since $Prime$ is obtained from primitive recursive relations using only primitive recursively bounded quantifiers and boolean operations, it follows that $Prime$ is primitive recursive. □

We will briefly expand on the details one last time before moving over to such arguments in the future. As in the previous argument, the set $\{(n, k, m) \in \mathbb{N}^3 : k \cdot m = n\}$ is primitive recursive. Using Proposition 12.1.20 twice in succession, we conclude that $\{n \in \mathbb{N} : (\exists k < n)(\exists m < n)[k \cdot m = n]\}$ is primitive recursive. Next, notice that $\{n \in \mathbb{N} : n > 1\}$ is a cofinite set, so is primitive recursive by Proposition 12.1.15 (or alternatively one can use the fact that Gt from Proposition 12.1.13 is primitive recursive). Finally, since the primitive recursive relations are closed under complement and intersection by Proposition 12.1.15, it follows that $Prime$ is primitive recursive.

We end this section with two other closure properties of primitive recursive functions. The first allows us to define a primitive recursive function using cases.

Proposition 12.1.23. *Suppose that $R_1, R_2, \ldots, R_n \subseteq \mathbb{N}^n$ are pairwise disjoint primitive recursive relations and that $g_1, g_2, \ldots, g_n, h \colon \mathbb{N}^n \to \mathbb{N}$ are primitive recursive functions. The function $f \colon \mathbb{N}^n \to \mathbb{N}$ defined by*

$$f(\vec{x}) = \begin{cases} g_1(\vec{x}) & \text{if } R_1(\vec{x}) \\ g_2(\vec{x}) & \text{if } R_2(\vec{x}) \\ \quad\vdots & \\ g_n(\vec{x}) & \text{if } R_n(\vec{x}) \\ h(\vec{x}) & \text{otherwise} \end{cases}$$

is primitive recursive.

Proof. Let $S = \mathbb{N}^n \backslash (R_1 \cup R_2 \cup \cdots \cup R_n)$, and note that S is a primitive recursive relation by Proposition 12.1.15. Since

$$f(\vec{x}) = g_1(\vec{x}) \cdot \chi_{R_1}(\vec{x}) + \cdots + g_n(\vec{x}) \cdot \chi_{R_n}(\vec{x}) + h(\vec{x}) \cdot \chi_S(\vec{x}),$$

for all $\vec{x} \in \mathbb{N}^n$, and both $+$ and $\cdot$ are primitive recursive, it follows that f is primitive recursive. □

In the paragraph preceding Proposition 12.1.20, we described the result informally as saying that the primitive recursive relations were closed under primitive recursively bounded search. In that context, the search resulted in a value of either 0 or 1, depending on whether the search found a witness y (or determined

that all values were witnesses, depending on the quantifier) such that $R(\vec{x}, y)$. Instead, we can think about searching for the first value of y below the bound such that $R(\vec{x}, y)$, and then outputting that value of y. If such a y does not exist, we can output the bound itself as a value indicating failure. We summarize this idea in the following definition.

Definition 12.1.24. *Let $n \in \mathbb{N}$, let $g\colon \mathbb{N}^{n+1} \to \mathbb{N}$, and let $R \subseteq \mathbb{N}^{n+1}$. If we define $f\colon \mathbb{N}^{n+1} \to \mathbb{N}$ by letting*

$$f(\vec{x}, y) = \begin{cases} w & \text{if there exists } z < g(\vec{x}, y) \text{ with } R(\vec{x}, z), \text{ and } w \text{ is the least such } z \\ g(\vec{x}, y) & \text{otherwise,} \end{cases}$$

then we use the notation

$$f(\vec{x}, y) = (\mu z < g(\vec{x}, y))R(\vec{x}, z).$$

Proposition 12.1.25. *Let $n \in \mathbb{N}$, let $g\colon \mathbb{N}^{n+1} \to \mathbb{N}$ be a primitive recursive function, and let $R \subseteq \mathbb{N}^{n+1}$ be a primitive recursive relation. The function $f\colon \mathbb{N}^{n+1} \to \mathbb{N}$ defined by*

$$f(\vec{x}, y) = (\mu z < g(\vec{x}, y))R(\vec{x}, z)$$

is primitive recursive.

It is helpful to read μ as "the least". Thus, $f(\vec{x}, y)$ is "the least $z < g(\vec{x}, y)$ such that $R(\vec{x}, z)$", unless no such z exists. Before diving into the proof, we sketch the key part of the argument. Given $\vec{x} \in \mathbb{N}^n$, how do we tell whether w is indeed the smallest value such that $R(\vec{x}, w)$? We want to know if $R(\vec{x}, w)$ and also that $\neg R(\vec{x}, z)$ for all $z < w$. Thinking computationally, the most direct way to determine this is to compute the value

$$\chi_R(\vec{x}, w) \cdot IsZero\left(\sum_{z<w} \chi_R(\vec{x}, z)\right).$$

If this value is 1, then w is the smallest values such that $R(\vec{x}, w)$. Otherwise, this value is 0.

Proof. Let

$$T = \{(\vec{x}, w) \in \mathbb{N}^{n+1} : R(\vec{x}, w) \text{ and } \neg R(\vec{x}, z) \text{ for all } z < w\},$$

i.e. $T(\vec{x}, w)$ if and only if w is the least value such that $R(\vec{x}, w)$. Notice that

$$\chi_T(\vec{x}, w) = \chi_R(\vec{x}, w) \cdot IsZero\left(\sum_{z<w} \chi_R(\vec{x}, z)\right)$$

for all $(\vec{x}, w) \in \mathbb{N}^{n+1}$, so T is primitive recursive by Proposition 12.1.18 and Proposition 12.1.12. Next, define $h\colon \mathbb{N}^{n+1} \to \mathbb{N}$ by letting

$$h(\vec{x}, y) = \sum_{z<g(\vec{x},y)} z \cdot \chi_T(\vec{x}, z).$$

Now h is primitive recursive by Corollary 12.1.19 together with the fact that the function $(\vec{x}, z) \mapsto z \cdot \chi_T(\vec{x}, z)$ is primitive recursive. Notice that for all $(\vec{x}, y) \in \mathbb{N}^{n+1}$, if there exists $z < g(\vec{x}, y)$ with $R(\vec{x}, z)$, and if w is the least such z, then $h(\vec{x}, y) = w$.

Next, let

$$S = \{(\vec{x}, y) \in \mathbb{N}^{n+1} : (\exists z < g(\vec{x}, y))R(\vec{x}, z)\},$$

and note that S is primitive recursive by Proposition 12.1.20. Since

$$f(\vec{x}, y) = \begin{cases} h(\vec{x}, y) & \text{if } S(\vec{x}, y) \\ g(\vec{x}, y) & \text{otherwise,} \end{cases}$$

we can use Proposition 12.1.23 to conclude that f is primitive recursive. □

We end this section with an example of how Proposition 12.1.25 (together with the other results in this section) can be used to quickly argue that the function which enumerates the primes in order is primitive recursive. We will make extensive use of this function to code sequences in a primitive recursive fashion in the next section.

Proposition 12.1.26. *The function $f\colon \mathbb{N} \to \mathbb{N}$ which sends n to the $(n+1)^{st}$ prime (i.e. $f(0) = 2$, $f(1) = 3$, $f(2) = 5$, etc.) is primitive recursive.*

Proof. Notice that by Euclid's proof that there are infinitely many primes, we have that $f(n+1) \leq f(n)!+1$ for all $n \in \mathbb{N}$. It follows that

$$\begin{aligned} f(0) &= 2 \\ f(n+1) &= (\mu z < (f(n)! + 2))(z > f(n) \wedge z \in Prime) \end{aligned}$$

for all $n \in \mathbb{N}$. Since the factorial function is primitive recursive by Proposition 12.1.9, the relation $>$ is primitive recursive by 12.1.13, the set $Prime$ is primitive recursive by Proposition 12.1.22, and the primitive recursive relation are closed under boolean connectives by Proposition 12.1.15, we can use Proposition 12.1.25 and Proposition 12.1.8 to conclude that f is primitive recursive. □

More formally, we are defining $h\colon \mathbb{N}^2 \to \mathbb{N}$ by letting

$$h(n, a) = (\mu z < a! + 2)(z > a \wedge z \in Prime),$$

and noting that h is primitive recursive, and then using Proposition 12.1.8 in the above argument. However, we will write in the more informal style, and omit many of the references to the fundamental propositions in this section, going forward.

12.2 Coding Sequences and Primitive Recursive Functions

We have developed the theory of primitive recursive functions as a theory only about functions f from some $\mathbb{N}^n$ to $\mathbb{N}$. In other words, we are only working with inputs and outputs that are natural numbers. What about integers, rational numbers, sequences, or other data types? Doesn't the restriction to natural numbers limit the scope of the kinds of "computation" that we can talk about? In this section, we show that we can code, in a primitive recursive manner, these more complex concepts using only natural numbers.

From a modern programming perspective, the ability to encode complicated data as numbers is not surprising, as data in a computer is stored as a sequence of zeros and ones, which can be thought of as a natural number when written in binary. However, this coding ability really is a revolutionary idea when pushed to its full potential. As we will see, we can even primitively recursively encode the primitive recursive functions using only natural numbers, which leads to some deep conclusions about the nature of computation.

We start with a discussion of how we can code a pair of natural numbers using just one natural number. Of course, we know that $\mathbb{N}^2$ is countable, so there is a bijection from $\mathbb{N}^2$ to $\mathbb{N}$. Is there a primitive recursive such bijection? Perhaps surprisingly, the usual bijection that walks down the finite diagonals of $\mathbb{N}^2$ is primitive recursive. The corresponding function $f\colon \mathbb{N}^2 \to \mathbb{N}$ begins as follows:

$$f(0,0) = 0, \quad f(0,1) = 1, \quad f(1,0) = 2 \quad f(0,2) = 3, \quad f(1,1) = 4, \quad f(2,0) = 5, \quad f(3,0) = 6, \quad \ldots$$

Can we describe f using a simple formula? Notice that first encode the pairs that sum to 0, i.e. just the pair $(0,0)$, and there is only 1 such pair. Next, we encode the 2 pairs that sum to 1 which are $(0,1)$ and $(1,0)$. After that, we encode the 3 pairs that sum to 2. Notice that in general, there are $n+1$ many pairs that sum to n. Thus, when thinking about the value of $f(x,y)$, we note that there are

$$\sum_{i=1}^{x+y} i$$

many pairs that sum to a value strictly less than $x+y$. We then order then $x+y+1$ many pairs that sum to $x+y$ in terms of increasing values of x, which leads to the conclusion that

$$f(x,y) = \left(\sum_{i=1}^{x+y} i\right) + x = \frac{(x+y)(x+y+1)}{2} + x,$$

which simplifies to

$$f(x,y) = \frac{x^2+y^2+2xy+3x+y}{2}.$$

The function $(x,y) \mapsto x^2+y^2+2xy+3x+y$ is primitive recursive because addition and multiplication are, but how do we deal with division by 2? Using the tools from the previous section, perhaps the easiest approach is to see that $f(x,y)$ is the least z with the property that $2z = x^2+y^2+2xy+3x+y$. Since this z is certainly less than or equal to $x^2+y^2+2xy+3x+y$, it follows that

$$f(x,y) = (\mu z < (x^2+y^2+2xy+3x+y+1))(2z = x^2+y^2+2xy+3x+y).$$

Therefore, f is primitive recursive by Proposition 12.1.25.

With the coding direction $f\colon \mathbb{N}^2 \to \mathbb{N}$ taken care of, we next turn our attention to decoding. Of course, the inverse of f is a function from $\mathbb{N}$ to $\mathbb{N}^2$, which is outside our domain of discourse when talking about primitive recursive functions. However, we can talk about the two "pieces" of this inverse function, i.e. the functions $g, h\colon \mathbb{N} \to \mathbb{N}$ such that $g(f(x,y)) = x$ and $h(f(x,y)) = y$ for all $x, y \in \mathbb{N}$. That is, $g(z)$ is the first component of the pair coded by z, and $h(z)$ is the second component of the pair coded by z. In order to easily argue that g and h are primitive recursive, the key observation is that $x \leq f(x,y)$ and $y \leq f(x,y)$ for all $x, y \in \mathbb{N}$. Thus, $g(z) \leq z$ and $h(z) \leq z$ for all $z \in \mathbb{N}$, from which it follows that

$$g(z) = (\mu x < z+1)[(\exists y < z+1)(f(x,y) = z)]$$

and

$$h(z) = (\mu y < z+1)[(\exists x < z+1)(f(x,y) = z)].$$

Since f is primitive recursive, we can therefore use Proposition 12.1.20 and Proposition 12.1.25 to conclude that the functions g and h are primitive recursive.

We can use the above primitive recursive function $f\colon \mathbb{N}^2 \to \mathbb{N}$ that codes pairs to create a primitive recursive function $f_3\colon \mathbb{N}^3 \to \mathbb{N}$ that codes triples as natural numbers. As described in Section 9.4 when we first considered using ordered pairs to define ordered triples in set theory, we can simply define $f_3(x,y,z) = f(f(x,y),z)$. Although this does work, and can be iterated to code n-tuples as numbers in a primitive recursive way, the problem is that one number now codes both a pair and a triple (and in fact an n-tuple for each n) simultaneously. Instead, we define one primitive coding of all finite sequences of natural numbers that avoids these issues. The idea is to use the uniqueness of prime factorizations. For example, to code the sequence $(7,1,4)$, we could use the elements of the sequence as the exponents of the first three primes. However, such a coding would not distinguish between $(7,1,4)$ and $(7,1,4,0)$. To avoid this problem, we simply add 1 to each element of the sequence when forming the exponents.

Definition 12.2.1. *For each $n \in \mathbb{N}^+$, let $\pi_n\colon \mathbb{N}^n \to \mathbb{N}$ be the function*

$$\pi_n(x_1, x_2, \ldots, x_n) = p_0^{x_1+1} p_1^{x_2+1} \cdots p_{n-1}^{x_n+1},$$

where $p_0, p_1, \ldots$ is the sequence of primes in ascending order.

Since multiplication and exponentiation are both primitive recursive, we immediately obtain the following.

Proposition 12.2.2. *For each $n \in \mathbb{N}^+$, the function π_n is primitive recursive.*

Notice that we do not need to use Proposition 12.1.26 here, since for each fixed n, the function π_n only refers to the first n primes. However, when thinking about decoding, we will often use the fact that the sequence of primes is primitive recursive. Before jumping into that, we first have to deal with the fact that not every number is actually a code for a sequence. For instance, $40 = 2^3 \cdot 3^0 \cdot 5^1$ does not code a sequence. Using 1 as a natural code for the empty sequence, we arrive at the following set.

Definition 12.2.3. *Let* $Seq = \{1\} \cup \bigcup_{n \in \mathbb{N}^+} range(\pi_n)$.

The following fact, which says that we can primitive recursively determine whether a number is a code for a sequence, is essential.

Proposition 12.2.4. *Seq is primitive recursive.*

Proof. Notice that $x \in Seq$ if and only if both $x > 0$, and whenever p is a prime that divides x, then every prime less than p also divides x. We do not need to quantify over all primes p in this description, since if p divides x, then $p \leq x$. Therefore, we have

$$x \in Seq \Leftrightarrow x > 0 \wedge (\forall p < x + 1)[(p \in Prime \wedge Divides(p, x)) \rightarrow (\forall q < p)(q \in Prime \rightarrow Divides(q, x))].$$

Since *Divides* and *Prime* are both primitive recursive by Proposition 12.1.21 and Proposition 12.1.22, and the primitive recursive relations are closed under boolean operations (noting that we can rewrite each instance of $\varphi \rightarrow \psi$ as $(\neg\varphi) \vee \psi$), bounded quantification, and bounded search, we conclude that *Seq* is primitive recursive. □

We now introduce some uniform notation for encoding sequences which avoids the dependence on the length.

Notation 12.2.5. *Given* $x_1, x_2, \ldots, x_n \in \mathbb{N}$, *we use* $\langle x_1, x_2, \ldots, x_n \rangle$ *to denote* $\pi_n(x_1, x_2, \ldots, x_n)$.

For example, we have $\langle 3, 2, 0 \rangle = 2^4 \cdot 3^3 \cdot 5^1 = 2160$. Now that we have established that the set of codes is primitive recursive, and that we can perform the encoding using primitive recursive functions, we turn our attention to decoding. Given a number $z \in Seq$, such as 2160, we would like to be able to determine the components of the sequence it codes, as well as its length, in a primitive recursive way. Since these values are related to the exponents that appear in the prime factorization of z, we first examine with the following function.

Proposition 12.2.6. *Define a function* $LargestPower \colon \mathbb{N}^2 \rightarrow \mathbb{N}$ *by letting* $LargestPower(x, y)$ *be the largest* $e \in \mathbb{N}$ *such that* x^e *divides* y *(here we define* $LargestPower(x, y) = 0$ *if either* $x = 1$ *or* $y = 0$, *which are the cases where no such largest* e *exists). The function LargestPower is primitive recursive.*

In the case where $x \neq 1$ and $y \neq 0$, the idea is to search for the smallest e such that x^e does not divide y, and then subtract one from the result.

Proof. Notice that if $x > 1$ and $y > 0$, then $x^y \geq 2^y > y$ and so x^y does not divide y. Also, if $x = 0$ and $y > 0$, then $x^y = 0$ and hence x^y does not divide y. Therefore, in all cases where $x \neq 1$ and $y \neq 0$, we know that x^y does not divide y. Using the notation of Proposition 12.1.13 and Proposition 12.1.10, it follows that

$$LargestPower(x, y) = Neq(x, 1) \cdot Neq(y, 0) \cdot Pred((\mu e < y + 1)(\neg Divides(x^e, y))).$$

Therefore, *LargestPower* is primitive recursive. □

Proposition 12.2.7. *Define a function* $DecodeSeq \colon \mathbb{N}^2 \rightarrow \mathbb{N}$ *defined by letting* $DecodeSeq(z, i)$ *be the* $(i + 1)^{st}$ *element of the sequence coded by* z *(and* 0 *if either* z *not code a sequence, or does not code a sufficiently long sequence). The function DecodeSeq is primitive recursive.*

Proof. Recall from Proposition 12.1.26 that the function $f(n) = p_n$ sending n to the $(n+1)^{st}$ prime is primitive recursive. Since

$$DecodeSeq(z,i) = \chi_{Seq}(z) \cdot Pred(LargestPower(p_i, z)).$$

it follows from Proposition 12.2.4 and Proposition 12.2.6 that *DecodeSeq* is primitive recursive. □

Since we will use the *DecodeSeq* often, we introduce some more compact notation that matches up with array notation in several programming languages.

Notation 12.2.8. *Given $z, i \in \mathbb{N}$, we use $z[i]$ to denote $DecodeSeq(z,i)$.*

Proposition 12.2.9. *Define a function $Len\colon \mathbb{N} \to \mathbb{N}$ by letting $Len(z)$ be the length of the sequence coded by z, if $z \in Seq$, and letting $Len(z) = 0$ otherwise. The function Len is primitive recursive.*

Proof. Notice that if $z \in Seq$, then the length of the corresponding sequence is simply the least i such that p_i does not divide z. Furthermore, since $p_z > z$, we can bound the size of the least such i. It follows that

$$Len(z) = \chi_{Seq}(z) \cdot (\mu i < z+1)(\neg Divides(p_i, z)),$$

and hence *Len* is primitive recursive. □

We have now handled all of the basic results related to encoding and decoding sequences. Next we examine some simple sequence operations, and verify that the corresponding functions are primitive recursive. We start with the following simple function.

Proposition 12.2.10. *Define a function $Append\colon \mathbb{N}^2 \to \mathbb{N}$ as follows. If $z \in Seq$ and $k \in \mathbb{N}$, then $Append(z,k)$ is the number that codes the sequence obtained by adding the number k onto the end of the sequence coded by z. If $z \notin Seq$, then $Append(z,k) = 0$. The function Append is primitive recursive.*

For example, we have $Append(\langle 3,0\rangle, 1) = \langle 3,0,1\rangle$. In terms of the underlying numbers, this can be restated as saying that $Append(48,1) = 1200$. From a computational point of view, if $z \in Seq$, then to compute $Append(z,k)$, we just need to multiply z by a suitable power of the next available prime, which is $p_{Len(z)}$. By using our developed tools, this leads to the following simple argument.

Proof. Notice that

$$Append(z,k) = \chi_{Seq}(z) \cdot z \cdot p_{Len(z)}^{k+1}$$

for all $z, k \in \mathbb{N}$. Therefore, *Append* is primitive recursive by Proposition 12.2.4, Proposition 12.1.26, and Proposition 12.2.9. □

The definition of primitive recursive functions includes, as its primary component, the ability to do step recursion on the natural numbers. However, step recursion alone does not directly allow more complicated constructions like the one used to define the Fibonacci numbers. To handle these, we needed to work with order recursion on the natural numbers, first described in Theorem 2.1.5, eventually formalized in Theorem 9.5.8, and then generalized to allow parameters in Theorem 9.5.9. Notice that each of these formalizations required the use of a sequence as an input in order to package up the history of previous values of the function. Although we can not use the sequences directly now, we can instead use the natural numbers that code these sequences. Given a function f, we now define a new function $\hat{f}$ that performs this packaging of values into a number.

Definition 12.2.11. *Let $n \in \mathbb{N}$. Given a function $f\colon \mathbb{N}^{n+1} \to \mathbb{N}$, we let $\hat{f}\colon \mathbb{N}^{n+1} \to \mathbb{N}$ be the function defined by*

$$\hat{f}(\vec{x}, y) = \langle f(\vec{x}, 0), f(\vec{x}, 1), \ldots, f(\vec{x}, y-1)\rangle.$$

In other words, $\hat{f}$ leaves the first n arguments of $\hat{f}$ alone (think of think of them as parameters), but codes up the history of values of f along the last argument y into a sequence or length y. This history is sometimes called the "course-of-values" of f.

Proposition 12.2.12. *Let $n \in \mathbb{N}$, and let $f\colon \mathbb{N}^{n+1} \to \mathbb{N}$. We have that f is primitive recursive if and only $\hat{f}$ is primitive recursive.*

Proof. If $\hat{f}$ is primitive recursive, then since $f(\vec{x}, y) = \hat{f}(\vec{x}, y+1)[y]$, we can use Proposition 12.2.7 to conclude that f is primitive recursive. Conversely, suppose f is primitive recursive. Notice that

$$\begin{aligned} \hat{f}(\vec{x}, 0) &= 1 \\ \hat{f}(\vec{x}, y+1) &= \mathit{Append}(\hat{f}(\vec{x}, y), f(\vec{x}, y)) \end{aligned}$$

for all $\vec{x} \in \mathbb{N}^n$ and $y \in \mathbb{N}$. Thus, defining $g\colon \mathbb{N}^n \to \mathbb{N}$ by $g(\vec{x}) = 0$ and $h\colon \mathbb{N}^{n+2} \to \mathbb{N}$ by $h(\vec{x}, y, a) = \mathit{Append}(a, f(\vec{x}, y))$, we have that $\hat{f} = \mathit{PrimRec}(g, h)$. Since g and h are primitive recursive, it follows that $\hat{f}$ is primitive recursive (note that we are assuming $n > 0$ here, but we can use Proposition 12.1.8 when $n = 0$). □

Using this coding of previous values of a function, we can now prove that the primitive recursive functions are closed under the more general concept of order recursion on $\mathbb{N}$. In other sources, this result is described as saying that the primitive recursive functions are closed under course-of-values recursion (rather than order recursion). Compare this result to Theorem 9.5.9, but where we allow multiple natural number parameters, rather than coding them as one n-tuple p.

Theorem 12.2.13. *Let $n \in \mathbb{N}$, and let $h\colon \mathbb{N}^{n+1} \to \mathbb{N}$ be primitive recursive. The function $f\colon \mathbb{N}^{n+1} \to \mathbb{N}$ defined by*

$$f(\vec{x}, y) = h(\vec{x}, \hat{f}(\vec{x}, y))$$

is primitive recursive.

Proof. Notice that we have

$$\begin{aligned} \hat{f}(\vec{x}, 0) &= 1 \\ \hat{f}(\vec{x}, y+1) &= \mathit{Append}(\hat{f}(\vec{x}, y), h(\vec{x}, \hat{f}(\vec{x}, y))) \end{aligned}$$

for all $\vec{x} \in \mathbb{N}^n$ and $y \in \mathbb{N}$. Thus, defining $g\colon \mathbb{N}^n \to \mathbb{N}$ by $g(\vec{x}) = 1$ and $h'\colon \mathbb{N}^{n+2} \to \mathbb{N}$ by $h'(\vec{x}, y, a) = \mathit{Append}(a, h(\vec{x}, a))$, we have that $\hat{f} = \mathit{PrimRec}(g, h')$. Since g and h' are primitive recursive, it follows that $\hat{f}$ is primitive recursive (as above, note that we are assuming $n > 0$ here, but we can use Proposition 12.1.8 when $n = 0$). Therefore, f is primitive recursive by Proposition 12.2.12. □

We can now use this result to argue that the Fibonacci sequence, defined by $f(0) = 0$, $f(1) = 1$, and $f(n) = f(n-1) + f(n-2)$ for all $n \geq 2$ is primitive recursive. We follow the argument after Theorem 2.1.5, but work with the codes of the sequence of previous values, rather than the sequence itself. Define $h\colon \mathbb{N} \to \mathbb{N}$ by letting

$$h(z) = \begin{cases} 0 & \text{if } z \in \mathit{Seq} \text{ and } \mathit{Len}(z) = 0 \\ 1 & \text{if } z \in \mathit{Seq} \text{ and } \mathit{Len}(z) = 1 \\ z[\mathit{Len}(z) - 2] + z[\mathit{Len}(z) - 1] & \text{if } z \in \mathit{Seq} \text{ and } \mathit{Len}(z) \geq 2 \\ 0 & \text{otherwise} \end{cases}$$

(here we are writing $-$ rather than $\dot{-}$ for slight ease of notation, and since it will be the actual $-$ in that case). Notice that h is primitive recursive by Proposition 12.1.23. Since $f(n) = h(\hat{f}(n))$ for all $n \in \mathbb{N}$, it follows from Theorem 12.2.13 that f is primitive recursive.

We can use sequences to code many other mathematical objects as natural numbers. For example, we can use sequences of zeros and ones to code finite graphs via their adjacency matrices (say, listed row by row in sequence). We will eventually use use our coding of sequences to code first-order formulas in each "reasonable" language as numbers, in such a way that the various operations on formulas are primitive recursive. However, we now turn our attention to a more surprising coding that gives insight into our attempt to formally define computation via the primitive recursive functions.

Every primitive recursive function is built up from the initial functions using finitely many applications of $Compose$ and $PrimRec$. For example, we saw earlier that + can be written as

$$PrimRec(\mathcal{I}_1^1, Compose(\mathcal{S}, \mathcal{I}_3^3)).$$

We can think of such an expression as instructions for how to calculate + in a very simple programming language. Now from a programming point of view, each program is a finite sequence of symbols from a finite alphabet, and hence can be coded as a number. If we apply that general idea here, we can code up each expression like $PrimRec(\mathcal{I}_1^1, Compose(\mathcal{S}, \mathcal{I}_3^3))$ with a number. There are many reasonable ways to perform such a coding, but the following recursive approach will best suit our purposes. If the reader has some experience in a programming language like LISP, it should feel familiar:

1. We use the number $\langle 0 \rangle = 2^{0+1} = 2$ as a code for the function $\mathcal{O}$.
2. We use the number $\langle 1 \rangle = 2^{1+1} = 4$ as a code for the function $\mathcal{S}$.
3. We use the number $\langle 2, n, i \rangle = 2^{2+1}3^{n+1}5^{i+1} = 8 \cdot 3^{n+1} \cdot 5^{i+1}$ as a code for the function $\mathcal{I}_i^n$ whenever $1 \leq i \leq n$.
4. If $a, b_1, b_2, \ldots, b_m$ are codes for primitive recursive functions such that the each of the functions coded by the b_i have the same arity, and the arity of the function coded by a is m, then we use the number $\langle 3, a, b_1, b_2, \ldots, b_m \rangle$ as a code for the function which is the composition of the function coded by a with the functions coded by the b_i.
5. If a and b are codes for primitive recursive functions such that the function coded by b has arity two more than the function coded by a, then we use the number $\langle 4, a, b \rangle$ as a code for the function with is achieved via primitive recursion using the function coded by a as the base case and the function coded by b as our iterator.

For example, the number

$$\langle 3, \langle 1 \rangle, \langle 0 \rangle \rangle = 2^{3+1} \cdot 3^{4+1} \cdot 5^{2+1} = 2^4 \cdot 3^5 \cdot 5^3 = 486,000$$

is a code for $Compose(\mathcal{S}, \mathcal{O})$, which is just the constant function $\mathcal{C}_1^1$. Notice that we are embedding sequence numbers as elements of other sequences recursively throughout our coding. As a result, these code numbers grow incredibly quickly. To get a taste of just how quickly, the code for $Compose(\mathcal{S}, Compose(\mathcal{S}, \mathcal{O}))$, which is the constant function $\mathcal{C}_2^1$, is

$$\langle 3, \langle 1 \rangle, \langle 3, \langle 1 \rangle, \langle 0 \rangle \rangle \rangle = 2^4 \cdot 3^5 \cdot 5^{2^4 \cdot 3^5 \cdot 5^3 + 1} = 3888 \cdot 5^{486,001}.$$

When we go just one more layer deep in the sequence coding, the numbers become astronomical. For example, we noted above that $PrimRec(\mathcal{I}_1^1, Compose(\mathcal{S}, \mathcal{I}_3^3))$ is just the function +, and the code for $PrimRec(\mathcal{I}_1^1, Compose(\mathcal{S}, \mathcal{I}_3^3))$ is $\langle 4, \langle 2, 1, 1 \rangle, \langle 3, \langle 1 \rangle, \langle 2, 3, 3 \rangle \rangle \rangle$. Now

$$\langle 2, 1, 1 \rangle = 2^3 \cdot 3^2 \cdot 5^2 = 1800$$

and

$$\langle 3, \langle 1 \rangle, \langle 2, 3, 3 \rangle \rangle = 2^4 \cdot 3^5 \cdot 5^{2^3 \cdot 3^4 \cdot 5^4} = 3888 \cdot 5^{405,001},$$

so our code is

$$\langle 4, \langle 2, 1, 1\rangle, \langle 3, \langle 1\rangle, \langle 2, 3, 3\rangle\rangle\rangle = 32 \cdot 3^{1,801} \cdot 5^{3888 \cdot 5^{405,001}+1}.$$

Although it is certainly possible to create other codings that grow at much more modest rate, we are only interested in highlighting the theoretical results that any such coding provides, so it is best to embrace the advantages of the simplicity of this approach (see below) and not to worry about the underlying numbers themselves.

It is very important to note that we are *not* actually coding primitive recursive functions directly as numbers, since a given primitive recursive function can be built from our initial functions using $Compose$ and $PrimRec$ in multiple ways. For example, although $2 = \langle 0\rangle$ is one code for the function $\mathcal{O}$, notice that $\langle 3, 2, 2\rangle = 54,000$ is a code for $Compose(\mathcal{O}, \mathcal{O})$, which is also the function $\mathcal{O}$.

Given a number $e \in \mathbb{N}$, how can we determine if it is a legitimate code? Let's first think it through from a naive computational point of view. We first check if either $e = \langle 0\rangle = 2$ or $e = \langle 1\rangle = 4$, in which case the answer is obviously yes. Otherwise, check if $e \in Seq$, and then check the value of $e[0]$. If $e[0] = 2$, then we need only check whether $Len(e) = 3$ and $1 \leq e[2] \leq e[1]$. However, we have a potential issue in the cases where either $e[0] = 3$ or $e[0] = 4$. Consider the simpler case where $e[0] = 4$. We can check whether $Len(e) = 3$ without an issue. We next check that $e[1]$ and $e[2]$ are each valid codes, which we can do using (order) recursion. The issue is that we then need to examine whether the arity of the function coded by $e[2]$ is two more than the arity of the function coded by $e[1]$. But if we are only keeping track of whether a number is or is not a valid code, then we do not have access to this information in the recursion.

To get around this problem, our function should put more information in the output than just whether it is a valid code. The relevant data that we need is the arity. Thus, we should define a function $f : \mathbb{N} \to \mathbb{N}$ where $f(e) = 0$ if e is not a valid code (since 0 is not a valid arity), and where $f(e) = n > 0$ if e is a valid code of a function of arity n. We would then solve the above issue, because we can recursively check whether $f(e[2]) = f(e[1]) + 2$, in which case we know that e is a valid code, and we then output $f(e) = f(e[1]) + 1$.

We now turn these ideas into a precise recursive definition, where we are using the fact that $e[i] < e$ for all $e, i \in \mathbb{N}$ with $e > 0$ in order to use order recursion.

Definition 12.2.14. *We define a function $f : \mathbb{N} \to \mathbb{N}$ recursively as follows:*

$$f(e) = \begin{cases} 1 & \text{if } e = \langle 0\rangle \\ 1 & \text{if } e = \langle 1\rangle \\ e[1] & \text{if } e \in Seq, e[0] = 2, Len(e) = 3, \text{ and } 1 \leq e[2] \leq e[1] \\ f(e[2]) & \text{if } e \in Seq, e[0] = 3, Len(e) \geq 3, (\forall i < Len(e))(i > 0 \to f(e[i]) \neq 0), \\ & \quad (\forall i < Len(e))(\forall j < Len(e))((i > 1 \wedge j > 1) \to f(e[i]) = f(e[j])), \\ & \quad \text{and } f(e[1]) = Len(e) - 2 \\ f(e[1]) + 1 & \text{if } e \in Seq, e[0] = 4, Len(e) = 3, f(e[1]) \neq 0, \text{ and } f(e[2]) = f(e[1]) + 2 \\ 0 & \text{otherwise.} \end{cases}$$

We denote this f by $PrimRecArity$.

The following fact essentially follows from Theorem 12.2.13 and Proposition 12.1.23 (which together say that a function defined recursively using primitive recursive conditions and functions will be primitive recursive), along with many of the other core closure properties of the primitive recursive functions that we have established.

Proposition 12.2.15. *$PrimRecArity$ is primitive recursive.*

Of course, technically we should write out the underlying function h that we are using in our appeal to Theorem 12.2.13 in order to verify that it is primitive recursive. We will do that once more here, but

nevermore, both to illustrate how it would be done, and also to convince the reader that they do not want to see such a thing again. So formally, we would define $h\colon \mathbb{N} \to \mathbb{N}$ by

$$h(z) = \begin{cases} 1 & \text{if } Len(z) = \langle 0 \rangle \\ 1 & \text{if } Len(z) = \langle 1 \rangle \\ Len(z)[1] & \text{if } Len(z) \in Seq, Len(z)[0] = 2, Len(Len(z)) = 3, \\ & \quad \text{and } 1 \leq Len(z)[2] \leq Len(z)[1] \\ z[Len(z)[2]] & \text{if } Len(z) \in Seq, Len(z)[0] = 3, Len(Len(z)) \geq 3, \\ & \quad (\forall i < Len(Len(z)))(i > 0 \to z[Len(z)[i]] \neq 0) \\ & \quad (\forall i < Len(Len(z)))(\forall j < Len(Len(z)))((i > 1 \wedge j > 1) \to \\ & \qquad z[Len(z)[i]] = z[Len(z)[j]], \\ & \quad \text{and } z[Len(z)[1]] = Len(Len(z)) - 2 \\ z[Len(z)[1]] + 1 & \text{if } Len(z) \in Seq, Len(z)[0] = 4, Len(Len(z)) = 3, z[Len(z)[1]] \neq 0, \\ & \quad \text{and } z[Len(z)[2]] = z[Len(z)[1]] + 2 \\ 0 & \text{otherwise.} \end{cases}$$

Next we would use the closure properties of the primitive recursive functions alluded to above to argue that h is primitive recursive, and finally we would check that $f(e) = h(\hat{f}(e))$ for all $e \in \mathbb{N}$ for the function f defined above, allowing us to use Theorem 12.2.13 to conclude that $f = PrimRecArity$ is primitive recursive.

We started our discussion of how we were going to code expression like $PrimRec(\mathcal{I}_1^1, Compose(\mathcal{S}, \mathcal{I}_3^3))$ with a high-level description, which then led to the question of how to tell if a number was a valid code. Now that we have formally defined our $PrimRecArity$, we can easily define the set of such valid codes.

Definition 12.2.16. *Let $PrimRecCode = \{e \in \mathbb{N} : PrimRecArity(e) > 0\}$. Also, for each $n \in \mathbb{N}^+$, let $PrimRecCode_n = \{e \in \mathbb{N} : PrimRecArity(e) = n\}$.*

Since $PrimRecArity$ is primitive recursive, we immediately obtain the following.

Proposition 12.2.17. *$PrimRecCode$ is primitive recursive, and for each $n \in \mathbb{N}^+$, the set $PrimRecCode_n$ is primitive recursive.*

We now define a "decoding function" Φ that turns inputs $e \in PrimRecCode$ into the primitive recursive functions that they code. Note that we are only defining it on elements of $PrimRecCode$, and we are using the fact (which follows from the recursive definition of $PrimRecCode$) that if $e \in PrimRecCode$ and $e[0] \in \{3, 4\}$, then $e[i] \in PrimRecCode$ whenever $0 < i < Len(e)$.

Definition 12.2.18. *Define a function $\Phi\colon PrimRecCode \to \mathcal{F}$ recursively as follows.*

$$\Phi(e) = \begin{cases} \mathcal{O} & \text{if } e = \langle 0 \rangle \\ \mathcal{S} & \text{if } e = \langle 1 \rangle \\ \mathcal{I}_{e[2]}^{e[1]} & \text{if } e[0] = 2 \\ Compose(\Phi(e[1]), \Phi(e[2])), \ldots, \Phi(e[Len(e) - 1]) & \text{if } e[0] = 3 \\ PrimRec(\Phi(e[1]), \Phi(e[2])) & \text{if } e[0] = 4. \end{cases}$$

The following result simply says that our definitions correctly code what they were designed to do. It can proven by induction on the generating system of primitive recursive functions in one direction, and by induction on e in the other.

Proposition 12.2.19. *Let $n \in \mathbb{N}^+$. A function $f\colon \mathbb{N}^n \to \mathbb{N}$ is primitive recursive if and only if there exists $e \in PrimRecCode_n$ such that $f = \Phi(e)$.*

We now return to the question of whether every intuitively computable function $f\colon \mathbb{N}^n \to \mathbb{N}$ is primitive recursive. Let's start with the much easier question of whether *every* function $f\colon \mathbb{N}^n \to \mathbb{N}$ is primitive recursive. In this case, the answer is clearly no, because there are uncountably many such functions but only countably many primitive recursive functions (either because the class of primitive recursive functions is generated from a countable collection of initial functions using countably many generated functions, or because we just coded them using natural numbers).

Let's examine the underlying argument here more closely without just immediately appealing to differences in cardinality. Since the collection of all primitive recursive functions is countable, we can list out all *unary* primary recursive functions, and then we can form a unary $f\colon \mathbb{N} \to \mathbb{N}$ that is not primitive recursive by simply diagonalizing out. But we can do more by making use of our above coding to *computably* diagonalize out of the primitive recursive functions! Here is how to do that in detail. Let's draw the standard two-dimensional grid used for diagonalization, and let the row numbered e code the unary function $\Phi(e)$ whenever $e \in PrimRecCode_1$. If $e \notin PrimRecCode_1$, we will just fill in the row with the unary function $\mathcal{O}$. Although a large portion of the rows of this table will be filled with zeros in our coding (including the first 1800 rows with the exception of line 4), we've filled this example in with some slightly more interesting values:

$\Phi(e)(x)$	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7
2	1	1	1	1	1	1	1	1
3	0	0	0	0	0	0	0	0
4	1	2	3	4	5	6	7	8
5	0	0	0	0	0	0	0	0
6	0	1	1	2	3	5	8	13
7	1	3	7	15	8	8	3	42

To recap, the entry in row e and column x is the value $\Phi(e)(x)$ whenever $e \in PrimRecCode_1$, and is 0 otherwise. Now to diagonalize out, we define $f\colon \mathbb{N} \to \mathbb{N}$ by letting

$$f(e) = \begin{cases} \Phi(e)(e) + 1 & \text{if } e \in PrimRecCode_1 \\ 1 & \text{otherwise,} \end{cases}$$

and note that f does not appear as any row of the above table, and so it is not primitive recursive. However, we can compute f! On input e, we first check whether $e \in PrimRecCode_1$, which we can do because we know that $PrimRecCode_1$ is primitive recursive, and hence certainly computable. If $e \notin PrimRecCode_1$, we simply output 1. Suppose then that $e \in PrimRecCode_1$. In this case, we take e, and break apart the code to form the underlying expression involving the initial functions and applications of $Compose$ and $PrimRec$. Next we work through the computation of what the resulting function does on input e, and finally simply add 1 to the result. Although annoying and not something that anybody would enjoy doing by hand, this is certainly a straightforward mechanical procedure that can be used to compute f. Therefore, there exists a computable unary function $f\colon \mathbb{N} \to \mathbb{N}$ that is not primitive recursive.

12.3 Partial Recursive Functions

Before moving on to a new definition that will include more of the class of "intuitively computable" functions, we should reflect on the argument at the end of the previous section. The diagonalization argument given there applies much more broadly than to just the primitive recursive functions. Although we embedded the argument within that setting in order to work out many of the details carefully, we can carry out the same argument to any class of unary functions $\mathcal{C}$ with the following properties:

1. Every element of $\mathcal{C}$ can be coded by a number (or by some finite object which can then be coded as a number) in such a way that we can decode that number and computably simulate the underlying function.

2. We can computably determine whether a given number is a valid code.

3. The functions in $\mathcal{C}$ provide an output for each input.

The first two properties are natural in light of our argument, but the third one might seem to have come out of nowhere. Note that we do certainly need the third condition in combination with the first one in order to perform a simulation *until* we find the value on a given argument, so that we can then change the resulting value as required in the diagonalization.

Let's examine the above properties from a modern programming point of view. A programming language unambiguously defines the grammar of a valid program. Since each program is a finite sequence of characters from a finite collection of allowable symbols, we can code programs by numbers, and we can simulate the execution of a program by stepping through the lines according to the control flow constructs of the language. Moreover, we can determine whether a given sequence of symbols satisfies the rules of the grammar, and hence can determine which numbers are valid codes (this is essentially what the lexer and parser do before the compiler turns the program into low level code). Therefore, modern programming languages satisfy the first two properties. However, they do not satisfy the third. It is certainly possible for a program to enter an infinite loop on some, or even all, inputs. In fact, if the reader has some experience with programming, then they have almost certainly created a program that ran forever despite that not being their intention. Thus, a program encodes a potentially *partial* function that does not provide an output for each input.

Since we need to give up at least one of the above three properties if we ever hope to define a class of functions that captures all intuitively computable functions, we choose to abandon the third one in light of the discussion in the previous paragraph. But before starting over and developing a formal programming language, we will first think about whether we can add something to our generating system for primitive recursive functions to accomplish the same task. Notice that the only obvious looping mechanism that we have in our definition is via a use of $PrimRec$. For example, suppose that we know how to compute a function $g\colon \mathbb{N} \to \mathbb{N}$ and we know how to compute a function $h\colon \mathbb{N}^3 \to \mathbb{N}$. Let's recall how we can compute $f = PrimRec(g, h)\colon \mathbb{N}^2 \to \mathbb{N}$. To determine $f(7, 5)$, we perform the following loop. First, compute the value $f(7, 0) = g(7)$. Next use this value to compute $f(7, 1) = h(7, 0, f(7, 0))$, then use this new value to compute $f(7, 2) = h(7, 1, f(7, 1))$, etc. In general, we start with $f(7, 0)$ and then loop a total of 5 times until we compute $f(7, 5)$. When we introduced this idea in the first section above, we noted that it resembled a simple for-loop in most programming languages, since we know in advance how many times will iterate through the loop (that number is simply the second argument).

Most programming languages allow a more flexible while-loop construction. The difference here is that we stay in the loop until a certain condition is satisfied, and we may not know in advance any upper bound on how many times we will iterate until that happens. In fact, we might end up stuck in the loop forever because the condition might not ever become true, which is one of the ways that a program can run forever. How can we incorporate this new more flexible idea into our framework? The key is to examine our bounded μ-operator. Recall that if $g\colon \mathbb{N}^{n+1} \to \mathbb{N}$ is a primitive recursive function and $R \subseteq \mathbb{N}^{n+1}$ is a primitive recursive relation, then Proposition 12.1.25 says that the function $f\colon \mathbb{N}^{n+1} \to \mathbb{N}$ defined by

$$f(\vec{x}, y) = (\mu z < g(\vec{x}, y))R(\vec{x}, z)$$

is primitive recursive. Note that we can think about computing f through a loop by checking $R(\vec{x}, 0)$, $R(\vec{x}, 1)$, etc. in turn until we either find a value of w with $R(\vec{x}, w)$, or we stop when we reach $g(\vec{x}, y)$. And like a direct application of $PrimRec$, we know in advance that we will loop at most $g(\vec{x}, y)$ many times. However, what if we allow such a search without any bound? That is, consider defining $f\colon \mathbb{N}^{n+1} \to \mathbb{N}$ by letting

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

i.e. $f(\vec{x}, y)$ is the least z such that $R(\vec{x}, z)$. When trying to implement this construct in a programming language, we would naturally use a while-loop, and exit only when we found the least such z. And if we implemented this while loop, then we would never exit if no such z exists. Thus, if we want to allow such a construction, we are naturally pushed in the direction of allowing partial functions that might not give an output on every input.

There are multiple ways to formally talk about partial functions. One natural approach is to say that a partial function from a set A to a set B is simply a relation $R \subseteq A \times B$ with the property that for every $a \in A$, there exists at most one $b \in B$ with $(a, b) \in R$. From this perspective, if we let

$$\text{domain}(R) = \{a \in A : \text{There exists } b \in B \text{ with } (a, b) \in R\},$$

then we obtain an honest function from domain(R) to B. Another approach is to introduce a new element to B, and define $f(a)$ to equal this new element when a fails to produce an output. Of course, we can easily go back and forth between these descriptions, but we will formally adopt the latter so that a partial function from A to B will be a literal function with domain A, and codomain equal to a set slightly larger than B.

Notation 12.3.1. *Fix some element not in $\mathbb{N}$ and denote it by $\uparrow$. Let $\mathbb{N}^{\uparrow} = \mathbb{N} \cup \{\uparrow\}$.*

Definition 12.3.2. *Let $\mathcal{F}$ be the set of all functions $\varphi\colon \mathbb{N}^n \to \mathbb{N}^{\uparrow}$ for some $n \in \mathbb{N}^+$. We call elements of $\mathcal{F}$* partial functions.

Notation 12.3.3. *Let $\varphi\colon \mathbb{N}^n \to \mathbb{N}^{\uparrow}$ be a partial function and let $\vec{x} \in \mathbb{N}$. We write $\varphi(\vec{x}) \uparrow$ to mean $\varphi(\vec{x}) = \uparrow$, and we write $\varphi(\vec{x}) \downarrow$ to mean that $\varphi(\vec{x}) \in \mathbb{N}$.*

Definition 12.3.4. *A partial function $\varphi\colon \mathbb{N}^n \to \mathbb{N}^{\uparrow}$ is called* total *if $\varphi(\vec{x}) \downarrow$ for all $\vec{x} \in \mathbb{N}^n$.*

We will typically use Greek letters like φ, ψ, θ, etc. to denote partial functions, and will reserve our traditional f, g, h, etc. for total functions. The notation $\uparrow$ is meant to evoke the image of a computation running forever, while $\downarrow$ represents that the computation eventually stops and produces an answer. Nevertheless, keep in mind that our objects are functions rather than instructions for how to perform a computation. That is, we will (for the moment) be working with static function objects rather than dynamic models of computation that evolve over time.

Our first task is to extend our definitions of *Compose* and *PrimRec* to the world of partial functions. Since inputs to partial functions are elements of $\mathbb{N}^n$ rather than elements of $(\mathbb{N}^{\uparrow})^n$, we need to take some some care here. However, in each case, we simply throw our hands in the air and output $\uparrow$ whenever one of the inputs is not an element of $\mathbb{N}$.

Definition 12.3.5. *Let $m, n \in \mathbb{N}^+$. Let $\varphi\colon \mathbb{N}^m \to \mathbb{N}^{\uparrow}$ and $\psi_1, \psi_2, \ldots, \psi_m\colon \mathbb{N}^n \to \mathbb{N}^{\uparrow}$ be partial functions. We let* Compose$(\varphi, \psi_1, \psi_2, \ldots, \psi_m)$ *be the partial function $\theta\colon \mathbb{N}^n \to \mathbb{N}^{\uparrow}$ defined by*

$$\theta(\vec{x}) = \begin{cases} \uparrow & \text{if } \psi_j(\vec{x}) \uparrow \text{ for some } j \\ \varphi(\psi_1(\vec{x}), \psi_2(\vec{x}), \ldots, \psi_m(\vec{x})) & \text{otherwise.} \end{cases}$$

Definition 12.3.6. *Let $n \in \mathbb{N}^+$. Let $\psi\colon \mathbb{N}^n \to \mathbb{N}^{\uparrow}$ and let $\varphi\colon \mathbb{N}^{n+2} \to \mathbb{N}^{\uparrow}$ be partial functions. We let* PrimRec(ψ, φ) *be the unique partial function $\theta\colon \mathbb{N}^{n+1} \to \mathbb{N}^{\uparrow}$ defined by*

$$\theta(\vec{x}, 0) = \psi(\vec{x})$$
$$\theta(\vec{x}, y+1) = \begin{cases} \uparrow & \text{if } \theta(\vec{x}, y) \uparrow \\ \varphi(\vec{x}, y, \theta(\vec{x}, y)) & \text{otherwise.} \end{cases}$$

We now introduce a new function into our generating system that performs a potentially unbounded search. In our above description, we assumed that we had a relation $R \subseteq \mathbb{N}^{n+1}$, and then we defined $f(\vec{x})$ to

be the least z such that $R(\vec{x}, z)$. However, we should define our basic operations on partial recursive functions rather than relations because those are our fundamental objects. Suppose then that $\psi\colon \mathbb{N}^{n+1} \to \mathbb{N}^{\uparrow}$ is a partial function. The idea is to define $\varphi\colon \mathbb{N}^n \to \mathbb{N}^{\uparrow}$ by letting $\varphi(\vec{x})$ be the least z such that $\psi(\vec{x}, z)$ has a certain property. The easiest such property is to let it be the least z such that $\psi(\vec{x}, z) = 0$, but note that this choice is in contrast to finding the least z with $\chi_R(\vec{x}, z) = 1$.

We need to grapple with one subtle point due to the potentially partial nature of ψ. Intuitively, to compute $\varphi(\vec{x})$, we would compute $\psi(\vec{x}, 0)$ to check if it equals 0, then compute $\psi(\vec{x}, 1)$ to check if it equals 0, etc. But what should happen if $\psi(\vec{x}, 0) = 7$, $\psi(\vec{x}, 1) \uparrow$, and $\psi(\vec{x}, 2) = 0$? From a computational point of view, if we follow the above idea of computing $\psi(\vec{x}, 0)$ to check if it equals 0, then move on to $\psi(\vec{x}, 1)$, etc., then our procedure will run forever and never reach the computation $\psi(\vec{x}, 2)$. Notice that this does match what would happen in a programming language with a simple while loop that proceeds through the values $0, 1, 2, \ldots$ in order and calls a subroutine to compute ψ on each in turn. As a result, we adopt the definition that defines $\varphi(\vec{x})$ to equal $\uparrow$ in this case. We thus arrive at the following definition.

Definition 12.3.7. *Let $n \in \mathbb{N}^+$ and let $\psi\colon \mathbb{N}^{n+1} \to \mathbb{N}^{\uparrow}$ be a partial function. We let* Minimize(ψ) *be the partial function $\theta\colon \mathbb{N}^n \to \mathbb{N}^{\uparrow}$ defined by*

$$\theta(\vec{x}) = \begin{cases} w & \text{if there exists } z \text{ with } \psi(\vec{x}, z) = 0 \text{ and } \psi(\vec{x}, u) \downarrow \text{ for all } u < z, \text{ and } w \text{ is the least such } z \\ \uparrow & \text{otherwise.} \end{cases}$$

We can now define a new class of partial functions using our generation template. We begin the same (total) initial functions that we used when defining primitive recursive functions, but we now add our new generating function $Minimize$, in addition to our more general versions of $Compose$ and $PrimRec$ that work on partial functions.

Definition 12.3.8. *The collection of* partial recursive functions *is the collection of partial functions generated by starting with the initial functions, and generating using Compose, PrimRec, and Minimize.*

Unfortunately, it is easy to confuse the names *partial recursive* function and *primitive recursive* function because the one word that differs starts with the same letter. We will still have occasion to work with primitive recursive functions, so it is important to keep that distinction. Sometimes, the class of partial recursive functions is called the class of *μ-recursive* functions, or the class of *general recursive* functions, or simply the class of *recursive* functions. We prefer to emphasize the word partial in the name, especially at this early stage. However, if the function happens to be total, then we will often use the following terminology that drops the word partial from the name.

Definition 12.3.9. *A* total recursive function *is a partial recursive function that is total, i.e. a partial recursive function $\varphi\colon \mathbb{N}^n \to \mathbb{N}^{\uparrow}$ such that $\varphi(\vec{x}) \downarrow$ for all $\vec{x} \in \mathbb{N}^n$.*

Since our more general versions of $Compose$ and $PrimRec$ behave identically to the old versions on total functions, we immediately obtain the following result.

Proposition 12.3.10. *Every primitive recursive function is a total recursive function.*

At this point, it is not clear whether the converse to this proposition is true. Of course, we hope that the added flexibility of $Minimize$ will allow us to argue that more functions are total recursive functions. For example, is the intuitively computable function described at the end of the last section (by computably diagonalizing out of the primitive recursive functions) a total recursive function? We will eventually see that the answer is yes.

Before jumping into such difficult questions, we begin with a much simpler one. Is there partial recursive function that is not total? Here is one simple way to answer this question. Recall from Proposition 12.1.6 that the constant function $\mathcal{C}_1^2$ which sends every pair (x, y) to the number 1 is primitive recursive. Thus, $\mathcal{C}_1^2$

is a total recursive function by Proposition 12.3.10. Let $\varphi = Minimize(f)$. We then have that $\varphi\colon \mathbb{N} \to \mathbb{N}^{\uparrow}$ is a partial recursive function. Since $\varphi(x)\uparrow$ for all $x \in \mathbb{N}$, the partial function φ is certainly not total.

For a more interesting example, let $R \subseteq \mathbb{N}^2$ be the relation

$$R = \{(x, y) \in \mathbb{N}^2 : x \text{ is even and } x < y\}.$$

Notice that x is even if and only if $Divides(2, x)$, so R is a primitive recursive relation by Proposition 12.1.21. Let $S = \mathbb{N}^2 \backslash R$. Since the primitive recursive relations are closed under complement, we know that S is a primitive recursive relation. Therefore, χ_S is a primitive recursive function, so is a total recursive function by Proposition 12.3.10, and we have

$$\chi_S(x, y) = \begin{cases} 0 & \text{if } x \text{ is even and } x < y \\ 1 & \text{otherwise.} \end{cases}$$

Now define $\varphi\colon \mathbb{N} \to \mathbb{N}^{\uparrow}$ by letting $\varphi = Minimize(\chi_S)$. We then have that φ is a partial recursive function and

$$\varphi(x) = \begin{cases} x + 1 & \text{if } x \text{ is even} \\ \uparrow & \text{otherwise.} \end{cases}$$

Since we have generalized the concept of primitive recursive functions to partial recursive functions, we can now use characteristic functions as above in order to define a potentially more general class of relations. Since characteristic functions are always total functions, we omit the words partial and total here, and simply use the word *recursive*.

Definition 12.3.11. *Let $R \subseteq \mathbb{N}^n$. We say that R is* recursive *if its characteristic function, i.e. the function $\chi_R\colon \mathbb{N}^n \to \mathbb{N}$ given by*

$$\chi_R(\vec{x}) = \begin{cases} 1 & \text{if } \vec{x} \in R \\ 0 & \text{otherwise,} \end{cases}$$

is a total recursive function.

Proposition 12.3.10 immediately gives us the following.

Proposition 12.3.12. *Every primitive recursive relation is a recursive relation.*

There do indeed exist recursive relations that are not primitive recursive, but we are not yet in a position to argue that (as we have not yet shown that there is a total recursive function that it is not primitive recursive). Before embarking on a deeper exploration of the total recursive functions and recursive relations, we first remark that these classes have the same fundamental closure properties as the partial recursive functions. We list all of these results in one theorem. The proofs of these results use exactly the same logic as the arguments in the primitive recursive case because everything is total and and our more general *Compose* and *PrimRec* return total functions on total inputs. As a result, we omit the essentially verbatim details.

Theorem 12.3.13. *We have the following:*

1. *If $R \subseteq \mathbb{N}^n$ is recursive, then $\mathbb{N}^n \backslash R$ is recursive.*
2. *If $R, S \subseteq \mathbb{N}^n$ are recursive, then $R \cap S$ is recursive.*
3. *If $R, S \subseteq \mathbb{N}^n$ are recursive, then $R \cup S$ is recursive.*

4. *If $n \in \mathbb{N}$, and $f, g\colon \mathbb{N}^{n+1} \to \mathbb{N}$ are both total recursive functions, then the functions $f', f''\colon \mathbb{N}^{n+1} \to \mathbb{N}$ defined by*
$$f'(\vec{x}, y) = \sum_{z<g(\vec{x},y)} f(\vec{x}, z) \qquad f''(\vec{x}, y) = \prod_{z<g(\vec{x},y)} f(\vec{x}, z)$$
are both total recursive functions.

5. *If $n \in \mathbb{N}$, $g\colon \mathbb{N}^{n+1} \to \mathbb{N}$ is a total recursive function, and $R \subseteq \mathbb{N}^{n+1}$ is recursive, then the sets*
$$S_1 = \{(\vec{x}, y) \in \mathbb{N}^{n+1} : (\exists z < g(\vec{x}, y))R(\vec{x}, z)\}$$
$$S_2 = \{(\vec{x}, y) \in \mathbb{N}^{n+1} : (\forall z < g(\vec{x}, y))R(\vec{x}, z)\}$$
are both recursive.

6. *Suppose that $n \in \mathbb{N}^+$, that $R_1, R_2, \ldots, R_n \subseteq \mathbb{N}^n$ are pairwise disjoint recursive relations and that $g_1, g_2, \ldots, g_n, h\colon \mathbb{N}^n \to \mathbb{N}$ are total recursive functions. The function $f\colon \mathbb{N}^n \to \mathbb{N}$ defined by*
$$f(\vec{x}) = \begin{cases} g_1(\vec{x}) & \text{if } R_1(\vec{x}) \\ g_2(\vec{x}) & \text{if } R_2(\vec{x}) \\ \quad\vdots & \\ g_n(\vec{x}) & \text{if } R_n(\vec{x}) \\ h(\vec{x}) & \text{otherwise} \end{cases}$$
is a total recursive function.

7. *If $n \in \mathbb{N}$, $g\colon \mathbb{N}^{n+1} \to \mathbb{N}$ is a total recursive function, and $R \subseteq \mathbb{N}^{n+1}$ is recursive, then the function $f\colon \mathbb{N}^{n+1} \to \mathbb{N}$ defined by*
$$f(\vec{x}, y) = (\mu z < g(\vec{x}, y))R(\vec{x}, z)$$
is a total recursive function.

8. *If $n \in \mathbb{N}$ and $h\colon \mathbb{N}^{n+1} \to \mathbb{N}$ is a total recursive function, then the unique function $f\colon \mathbb{N}^{n+1} \to \mathbb{N}$ defined by*
$$f(\vec{x}, y) = h(\vec{x}, \hat{f}(\vec{x}, y))$$
is a total recursive function.

In several parts of this theorem, we can replace "total recursive" with "partial recursive" throughout the statement, but we then occasionally have to be careful with our interpretation of the result. Let's examine (4) in this context. Suppose instead that $n \in \mathbb{N}$ and $\varphi, \psi\colon \mathbb{N}^{n+1} \to \mathbb{N}^{\uparrow}$ are partial recursive functions. If we want to define $\varphi'\colon \mathbb{N}^{n+1} \to \mathbb{N}$ by
$$\varphi'(\vec{x}, y) = \sum_{z<\psi(\vec{x},y)} \varphi(\vec{x}, z),$$
then we need to be clear about how to handle situations where $\psi(\vec{x}, y)\uparrow$ or where $\psi(\vec{x}, y)\downarrow$ but there exists $z < \psi(\vec{x}, y)$ with $\varphi(\vec{x}, z)\uparrow$. If we naturally define $\varphi'(\vec{x}, y)\uparrow$ in any of these cases, then we can follow the proofs of Proposition 12.1.18 and Proposition 12.1.19 to argue that φ' is indeed a partial recursive function. We can similarly generalize (7) to the case where g is partial recursive by letting $f(\vec{x}, y)\uparrow$ whenever $g(\vec{x}, y)\uparrow$. The corresponding generalization of (6) to partial recursive functions also goes through by simply following the proof of Proposition 12.1.23. It is important enough that we state it here.

Proposition 12.3.14. *Suppose that $n \in \mathbb{N}^+$, that $R_1, R_2, \ldots, R_n \subseteq \mathbb{N}^n$ are pairwise disjoint recursive relations and that $\psi_1, \psi_2, \ldots, \psi_n, \theta \colon \mathbb{N}^n \to \mathbb{N}^\uparrow$ are partial recursive functions. The partial function $\varphi \colon \mathbb{N}^n \to \mathbb{N}^\uparrow$ defined by*

$$\varphi(\vec{x}) = \begin{cases} \psi_1(\vec{x}) & \text{if } R_1(\vec{x}) \\ \psi_2(\vec{x}) & \text{if } R_2(\vec{x}) \\ \quad\vdots & \\ \psi_n(\vec{x}) & \text{if } R_n(\vec{x}) \\ \theta(\vec{x}) & \text{otherwise} \end{cases}$$

is a partial recursive function.

However, it is not possible to easily generalize (5) and (8) to partial recursive functions. For (8), it's not clear how to even define $\hat{\varphi}$ when $\varphi \colon \mathbb{N} \to \mathbb{N}^\uparrow$ is a partial function because what should $\langle 2, \uparrow, 7 \rangle$ mean? We run into a much deeper issue with trying to generalize (5) to partial functions. If $g(\vec{x}, y) \uparrow$, then should we put $(\vec{x}, y)$ in S_1 or not? Now χ_{S_1} will be a total function regardless, but no matter which option we choose, there is no obvious way to computationally determine whether $g(\vec{x}, y) \uparrow$. As result, there will be no obvious procedure to compute χ_{S_1}. In fact, this wrinkle brings us face-to-face with the *halting problem*, a fundamental topic that we will return to in the Section 12.5.

Before moving on, we establish a couple of simple results about partial recursive functions. First, recall that when we were originally thinking about introducing an unbounded μ-operator, we motivated the idea in terms of a relation. That is, if $R \subseteq \mathbb{N}^{n+1}$ is recursive, then we wanted to define a function $f \colon \mathbb{N}^n \to \mathbb{N}$ by letting $f(\vec{x})$ be the least z such that $R(\vec{x}, z)$. We did not follow this idea in the definition of $Minimize$ because our fundamental objects are functions, but we can now argue the the resulting partial function is indeed partial recursive whenever R is recursive.

Proposition 12.3.15. *Let $n \in \mathbb{N}^+$ and $R \subseteq \mathbb{N}^{n+1}$ be a recursive relation. The partial function $\varphi \colon \mathbb{N}^n \to \mathbb{N}^\uparrow$ defined by letting $\varphi(\vec{x}) = \mu z R(\vec{x}, z)$, i.e.*

$$\varphi(\vec{x}) = \begin{cases} w & \text{if there exists } z \text{ with } R(\vec{x}, z), \text{ and } w \text{ is the least such } z \\ \uparrow & \text{otherwise,} \end{cases}$$

is a partial recursive function.

Proof. Let $S = \mathbb{N}^{n+1} \backslash R$, and notice that S is recursive because the recursive relations are closed under complement. Thus, by definition, χ_S is a total recursive function. Since $\varphi = Minimize(\chi_S)$, it follows that φ is a partial recursive function. □

Finally, we pause to note that the inclusion of unbounded search in our definition of partial recursive functions allows us to solve an issue alluded to before Proposition 12.1.17. In our new setting, we can prove the following equivalence.

Proposition 12.3.16. *Let $f \colon \mathbb{N}^n \to \mathbb{N}$ be a total function. We then have that f is a total recursive function if and only if $graph(f) \subseteq \mathbb{N}^{n+1}$ is a recursive relation.*

Proof. The left-to-right direction is identical to the proof of Proposition 12.1.17, so we prove the right-to-left direction. Suppose then that $graph(f)$ is a recursive relation. Since f is total, we know that for all $\vec{x} \in \mathbb{N}$, there exists a unique $z \in \mathbb{N}$ such that $(\vec{x}, z) \in graph(f)$, namely $z = f(\vec{x})$. Therefore, $f(\vec{x}) = \mu z((\vec{x}, z) \in graph(f))$, and hence f is a total recursive function by Proposition 12.3.15. □

We could now embark on a coding of the expressions formed by starting with the initial functions and applying $Compose$, $PrimRec$, and $Minimize$ which extends the coding of the previous section. For

example, we could simply add another clause that uses $\langle 5, a \rangle$ as a code to represent applying $Minimize$ to the expression coded by a, assuming that the underlying function has arity at least 2. With this idea, the details of the coding are straightforward, and so will be omitted here. In this setting, we would then have that $\Phi(e)$ is a partial recursive function for each e that is a valid code. Thus, we run into an issue if we try to computably diagonalize out since the underlying functions might result in $\uparrow$ on some inputs. If we attempted to simulate $\Phi(e)$ on some input, we seemingly have no way to know computationally whether the output is $\uparrow$. Thus, we once again run into the so-called *halting problem* that will be discussed in Section 12.5.

12.4 A Machine Model of Computation

In order to obtain a better intuitive understanding of the class of partial recursive functions, as well as the somewhat mysterious nature of partial recursive functions producing $\uparrow$, we now develop an idealized machine model of computation. Modern computers store data in memory, and proceed in stages to apply one of a finite set of transformations (coded by machine instructions) to the underlying data at each clock cycle. We will develop a very simple formal model in this vein, which can also be viewed as a tiny formal programming language, that provides us with a direct dynamic model of computation where state evolves over time.

The most widely used and celebrated machine model of computation is the *Turing machine*. This model was historically groundbreaking, both because Turing successfully argued at a high level that every "mechanical" procedure can be simulated by one of these machines, and also because the model served as a blueprint for some of the first real computing machines that were built in the middle of the 20^{th} century. Moreover, in his fundamental paper, Turing used his model to answer an important open question in mathematical logic that arose from Hilbert's Program.

Without going into all of the technical details, we give a brief description of Turing's model. A Turing machine starts with a finite alphabet. The machines has a read/write head that acts on an infinite tape which is divided into infinitely many cells, each of which contains a symbol from the finite alphabet. In order to describe how the head interacts with the tape, a Turing machine also has a finite set of possible states, along with a function that says what to do when the read/write head is in a given state and is reading a certain symbol. The function outputs what state to transition to, what to write on the corresponding cell, and also which direction to move the head.

The Turing machine model has many strengths. For example, it encodes inputs and outputs using finite strings of symbols from a finite alphabet, which is very natural from the perspective of computer science. In addition, Turing machines naturally provide a straightforward way to define the complexity of computation in terms of the amount of time and space used. However, there are a few weaknesses from our current perspective. First, it does take a good deal of time and effort to define everything carefully. Next, it is somewhat challenging to build Turing machines that compute simple functions, and it is even more challenging to argue carefully that the functions they compute are closed under various operations. Finally, since we are working with functions on the natural numbers, we would need to deal with representations of the natural numbers in some string format, such as writing them in unary, binary, decimal, etc.

Instead of following this path, we develop a simpler machine model of computation called the *unlimited register machine*, or URM. Idealized register machines were first introduced by Shepherdson and Sturgis, but we will follow a small variation that is used by Cutland in his book *Computability*. Like a Turing machine, this model has infinitely many cells, which in this setting are called *registers*. The primary advantage over Turing machines is that each register can store an arbitrarily large natural number (whereas the cells on a tape in a Turing machine can only store a character from the finite alphabet), hence the word "unlimited" in the name. A secondary advantage is that this model allows random access to the registers. That is, we can immediately examine the contents of a register far away without having to methodically move a read/write head across the tape in small incremental steps. The disadvantages of our model are that it takes more effort to believe that it encapsulates general purpose computation relative to Turing machines, and that it only works with natural numbers (so we will have to code other objects as numbers).

We now dive into the details. We envision an infinite sequence of registers, which we denote by $R_1, R_2, R_3, \ldots$. At any given stage, each R_i contains a natural number r_i.

Definition 12.4.1. *An* unlimited register machine*, or URM, consists of a finite sequence of instructions, each of which is of four possible types:*

- $Z(i)$ *for* $i \in \mathbb{N}^+$*: Change the value of the* i^{th} *register to* 0*, so set* $r_i = 0$*.*
- $S(i)$ *for* $i \in \mathbb{N}^+$*: Increment the value of the* i^{th} *register by* 1*, so change* r_i *to be the value* $r_i + 1$
- $T(i, j)$ *for* $i, j \in \mathbb{N}^+$*: Transfer the value of the* i^{th} *register into the* j^{th} *register, without changing the value of the* i^{th} *register. In other words, change the value of* r_j *to equal the value* r_i*.*
- $J(i, j, q)$ *for* $i, j, q \in \mathbb{N}^+$*: Check if the value of the* i^{th} *register is equal to the value of the* j^{th} *register. If so, then jump to the* q^{th} *instruction. Otherwise, do nothing.*

Note that the descriptions of the instructions after each colon are not part of the formal definition. A URM is simply a finite sequence of instructions. We will turn the intuitive descriptions of how the instructions "act" into a formal definition soon, but we first give an example of a URM and examine the corresponding dynamic computation. Consider the following URM, annotated with the instruction number in parentheses:

(1) $J(2, 3, 5)$
(2) $S(1)$
(3) $S(3)$
(4) $J(1, 1, 1)$

The idea is to start at the first instruction, perform the action described above, and then move on to the next instruction in line (unless we have a J instruction that jumps elsewhere). Now in order to perform a computation, we need to start with an initial sequence of values in the registers. Suppose then that we start in the state where $r_1 = 1$, $r_2 = 2$, and $r_i = 0$ for all $i > 2$. We denote this state by the infinite sequence $1, 2, 0, 0, \ldots$. Now since $r_2 \neq r_3$, the first instruction does nothing, and so leaves us in the state $1, 2, 0, 0, \ldots$. We move on to the second instruction, which increments the value of r_1 so that we are now in the state $2, 2, 0, 0$. We then move on to the third instruction, which changes the state to $2, 2, 1, 0, \ldots$. Now the fourth tests whether r_1 equals itself, which of course it always does, and so it causes a jump back to the first instruction. In other words, the fourth instruction says to GOTO instruction 1 in all circumstances. If we continue to follow the program, we obtain the following sequence of states, where the number in parentheses at the front is the next instruction to execute:

$$\begin{aligned}
(1) : 1, 2, 0, 0, \ldots &\mapsto (2) : 1, 2, 0, 0, \ldots \\
&\mapsto (3) : 2, 2, 0, 0, \ldots \\
&\mapsto (4) : 2, 2, 1, 0, \ldots \\
&\mapsto (1) : 2, 2, 1, 0, \ldots \\
&\mapsto (2) : 2, 2, 1, 0, \ldots \\
&\mapsto (3) : 3, 2, 1, 0, \ldots \\
&\mapsto (4) : 3, 2, 2, 0, \ldots \\
&\mapsto (1) : 3, 2, 2, 0, \ldots \\
&\mapsto (5) : 3, 2, 2, 0, \ldots
\end{aligned}$$

Since there is no fifth instruction, we need to define what to do next. Intuitively, we think of the machine as stopping at this point because there is no valid instruction to execute next.

We can now turn these ideas of a URM causing state to change over time into a definition of what function(s) a URM "computes". Given a URM $M = (I_1, I_2, \ldots, I_s)$ consisting of a sequence of s instructions, together with an $n \in \mathbb{N}^+$, we define a corresponding partial function from $\mathbb{N}^n$ to $\mathbb{N}^\uparrow$ as follows. On input $(x_1, x_2, \ldots, x_n)$, start the machine in the state where $r_i = x_i$ for all i with $1 \leq i \leq n$ and where $r_i = 0$ for all $i > n$. Follow the instructions as outlined above until we end up trying to execute an instruction $k > s$. In this case, we output the value r_1 that is currently in the first register. Otherwise, if we never try to execute an instruction $k > s$, then we output $\uparrow$. For example, with our above URM and $n = 2$, then our partial function outputs 3 on input $(1, 2)$.

Although we have provided a comprehensive description of how to take a URM and an $n \in \mathbb{N}^+$, and determine a corresponding partial function from $\mathbb{N}^n$ to $\mathbb{N}^\uparrow$, it will be useful to have a slightly more careful definition. To facilitate such a definition, we examine the concrete run of the example URM above. Note that we annotated each line with the next instruction. We can think of encoding line m of execution as a function $f_m \colon \mathbb{N} \to \mathbb{N}$, where $f_m(0)$ is the instruction that we should execute next, and where $f_m(j) = r_j$ equals the current contents of register R_j for all $j > 0$. We can then turn the intuitive descriptions of the actions of each of the instructions into rules for how to determine f_{m+1} from f_m. For example, if $f_m(0) = k \leq s$ and $I_k = Z(i)$, then

$$f_{m+1}(j) = \begin{cases} f_m(0) + 1 & \text{if } j = 0 \\ 0 & \text{if } j = i \\ f_m(j) & \text{otherwise.} \end{cases}$$

Notice that the first line here simply increments the value in position 0 of f_m, which reflects the fact that we should move on the next instruction in this situation. Similarly, we can define f_{m+1} in terms of f_m for the other three instructions. Now in the situation where $f_m(0) > s$, then we said above that we intuitively think that the machine stops at this point. Formally, we simply let $f_{m+1} = f_m$ so that we always define an infinite sequence of state functions. Putting everything together, we arrive at the following definition.

Definition 12.4.2. *Let $M = (I_1, I_2, \ldots, I_s)$ be a URM, and let $n \in \mathbb{N}^+$. We define a partial function $\varphi_M^{(n)} \colon \mathbb{N}^n \to \mathbb{N}^\uparrow$ as follows. On input $\vec{x} = (x_1, x_2, \ldots, x_n) \in \mathbb{N}^n$, let $f_1 \colon \mathbb{N} \to \mathbb{N}$ be the infinite sequence $(1, x_1, x_2, \ldots, x_n, 0, 0, 0, \ldots)$, and then recursively define a sequence $f_1, f_2, f_3, \ldots$ as outlined above. If there exists an m such that $f_m(0) > s$, then letting ℓ be the least such m, we define $\varphi_M^{(n)}(\vec{x}) = f_\ell(1)$. If no such m exists, then we define $\varphi_M^{(n)}(\vec{x}) \uparrow$.*

Thus, letting $M = (J(2, 3, 5), S(1), S(3), J(1, 1, 1))$, we have $\varphi_M^{(2)}(1, 2) = 3$. In general, the machine M repeatedly increments the contents of register 1 and 3 until register 3 has the value that starts in register 2. Thus, on inputs $(x_1, x_2, 0, 0, 0, \ldots)$, the machine will eventually jump to instruction 5 with state $(x_1 + x_2, x_2, x_2, 0, 0, \ldots)$. It follows that $\varphi_M^{(2)}(x_1, x_2) = x_1 + x_2$ for all $x_1, x_2 \in \mathbb{N}$. What happens when we consider an arity $n \neq 2$? Notice that $\varphi_M^{(1)}(x_1) = x_1$ for all $x_1 \in \mathbb{N}$ because we immediately jump to instruction 5. Thus, $\varphi_M^{(1)}$ is the identity function, which is just the initial function $\mathcal{I}_1^1$. In contrast, we have $\varphi_M^{(3)}(0, 0, 1) \uparrow$ because we will forever increment the contents of the registers 1 and 3 without ever reaching a place where $r_2 = r_3$.

These examples illustrate an essential point. A URM M is simply a sequence of instructions, so does not come equipped with an arity! We need to provide an arity $n \in \mathbb{N}^+$ in addition to M in order to define what partial function M computes. With this in mind, we now provide another definition for when a partial function is computable.

Definition 12.4.3. *Let $n \in \mathbb{N}^+$ and let $\psi \colon \mathbb{N}^n \to \mathbb{N}^\uparrow$ be a partial function. We say that ψ is* URM-computable *if there exists a URM M such that $\psi = \varphi_M^{(n)}$.*

For instance, the addition function $f \colon \mathbb{N}^2 \to \mathbb{N}$ is URM-computable because $f = \varphi_M^{(2)}$ for the machine M described above. For a more interesting example, consider the partial recursive function $\psi \colon \mathbb{N} \to \mathbb{N}^\uparrow$

described above that is given by

$$\psi(x) = \begin{cases} x+1 & \text{if } x \text{ is even} \\ \uparrow & \text{otherwise.} \end{cases}$$

To show that ψ is URM-computable, we simply need to exhibit a URM that computes it. Intuitively, we repeatedly increment the contents of the second register by 2 until, if ever, the contents of the first two registers agree. If so, we have determined that the input (the contents of the first register) is even, so we simply increment the first register and stop. Working out the details, we arrive at the following URM M:

(1) $J(1, 2, 5)$
(2) $S(2)$
(3) $S(2)$
(4) $J(1, 1, 1)$
(5) $S(1)$

It is then straightforward to check that $\psi = \varphi_M^{(1)}$.

If we instead start with a URM M, it can be quite difficult to "see" what the function $\varphi_M^{(n)}$ is. For example, consider the following URM M:

(1) $T(1, 5)$
(2) $Z(1)$
(3) $S(2)$
(4) $J(5, 6, 14)$
(5) $T(1, 3)$
(6) $T(2, 1)$
(7) $J(3, 4, 11)$
(8) $S(2)$
(9) $S(4)$
(10) $J(2, 2, 7)$
(11) $Z(4)$
(12) $S(6)$
(13) $J(1, 1, 4)$

What is $\varphi_M^{(1)}$? It turns out that $\varphi_M^{(1)}(n)$ equals the n^{th} Fibonacci number a_n defined by $a_0 = 0$, $a_1 = 1$, and $a_k = a_{k-1} + a_{k-2}$ for $k \geq 2$. To see this, we walk through the execution of the machine at a high level. On input n, the machine begins by setting r_5 equal to the input n, setting r_1 to $0 = a_0$, and setting r_2 to $1 = a_1$. The idea now is that we will loop through incrementing r_6 until it equals the initial input n that is stored in register 5, while repeatedly making the values of the first two registers store consecutive Fibonacci numbers. That is, assuming that $r_1 = a_k$ and $r_2 = a_{k+1}$ (as well as assuming that $r_4 = 0$) at the beginning of the loop, then after one iteration of the loop we will have $r_1 = a_{k+1}$ and $r_2 = a_{k+2}$ (as well as $r_4 = 0$). The instruction I_4 performs the check to see whether r_5 equals r_6, and ends execution in that case. The heart of the machine is the loop from I_5 to I_{13}. Instructions I_5 and I_6 copy the value a_k that we assume is stored in register 1 to register 3, and then copies the value a_{k+1} that is stored in register 2 to register 1. At this point, the first four registers have the values $a_{k+1}, a_{k+1}, a_k, 0$. Thus, we have successfully moved a_{k+1} into register 1. Now we want to change r_2 to equal $a_{k+2} = a_{k+1} + a_k$, so we simply want to add the values of r_2 and r_3, and put the result in register 2. Fortunately, we already have a URM that performs addition! We

insert that URM as a subroutine, but we shift the registers that it works on. In other words, instructions I_7 through I_{10} are simply the result of taking our URM for addition, adding 1 to each named register, and changing the location of the jump in I_{10} to be 7 rather than 1. After this addition is performed, the first four registers will be $a_{k+1}, a_{k+2}, a_k, a_k$, so we zero out register 4, increment the contents of register 6, and jump back to I_4 to continue the loop. As mentioned above, this loop will continue until r_6 equals the initial input n, at which point the value r_1 will equal a_n, as desired.

Our justification that the machine computes the n^{th} Fibonacci number, just like our justification that the previous two machines computed the outputs described, is somewhat informal. It is certainly possible to give a careful proof by stating and proving properties of the corresponding recursively defined functions f_m by induction on m. In this case, such an argument would be a detailed version of a loop invariant proof in computer science. However, these arguments are simultaneously painful and essentially routine using our high level descriptions, and do not provide much additional insight. Therefore, we will continue to use our more informal style.

We now observe a few simple facts about URMs that will be important in the coming arguments:

- Every URM is a finite sequence of instructions, each of which refers to at most two registers, so every URM refers to only finitely many registers. Therefore, if the initial state consists of only finitely many nonzero registers (as it does when we define any $\varphi_M^{(n)}$), then there are only finitely many registers that will have a nonzero value at any point during the machine's execution. More formally, in this case, there exists $k \in \mathbb{N}^+$ such that $f_m(i) = 0$ for all $m \in \mathbb{N}^+$ and all $i > k$.

- If $M = (I_1, I_2, \ldots, I_s)$ is a URM with s instructions, then we can assume that every jump instructions $J(i, j, q)$ of M has the property that $q \leq s + 1$. More formally, there exists a URM $N = (I_1, I_2, \ldots, I_s)$ with this property such that $\varphi_M^{(n)} = \varphi_N^{(n)}$ for all $n \in \mathbb{N}^+$. In order to construct N, we simply change all such q with $q > s + 1$ to be equal to $s + 1$. This property will be useful if we want to use M as a subroutine like we did for addition in our Fibonacci machine.

- Given a URM M, we can systematically shift the register numbers referred to in each instruction by the same amount in order to form a URM N that behaves the same as M, but operates on the shifted registers. For example, in our Fibonacci machine, we took the addition machine and shifted every register by 1, to result in the instructions I_7 through I_{10} that performed addition using registers 2 through 4 rather than registers 1 through 3.

With these facts in hand, we can prove our first major theorem about URM-computable functions.

Theorem 12.4.4. *Every partial recursive function is URM-computable.*

Proof. The proof is by induction on the generation of the partial recursive functions. Thus, we show that the initial functions are URM-computable, and that the URM-computable functions are closed under $Compose$, $PrimRec$, and $Minimize$.

- Initial Functions: Letting $M = (Z(1))$, we have $\mathcal{O} = \varphi_M^{(1)}$. Similarly, letting $M = (S(1))$, we have $\mathcal{S} = \varphi_M^{(1)}$. Finally, given $i, n \in \mathbb{N}^+$ with $1 \leq i \leq n$, if we let $M = (T(i, 1))$, then $\mathcal{I}_i^n = \varphi_M^{(n)}$.

- Closure under Composition: Suppose that $\varphi\colon \mathbb{N}^m \to \mathbb{N}^\uparrow$ and $\psi_1, \psi_2, \ldots, \psi_m\colon \mathbb{N}^n \to \mathbb{N}^\uparrow$ are all URM-computable. We show that $\theta = Compose(\varphi, \psi_1, \psi_2, \ldots, \psi_m)$ is URM-computable. Fix a URM M that computes φ (i.e. such that $\varphi = \varphi_M^{(m)}$), and fix URMs N_i that compute each ψ_i. Let k be the maximum number of any register referred to in the machines $M, N_1, N_2, \ldots, N_m$. We describe a URM that computes θ.

  On input $\vec{x} \in \mathbb{N}^n$, we copy the values of the registers $1, 2, \ldots, n$ to the registers $n+1, n+2, \ldots, 2n$, and also to $2n+1, 2n+2, \ldots, 3n$. Next we insert a modified version of N_1 where each register is shifted by

$2n$ (and jump instructions are shifted accordingly), which ends with the value $\psi_1(\vec{x})$ in register $2n+1$. We then transfer this value to register 1, and zero out registers $3n+1$ through $\max\{3n+1, 2n+k\}$. Next we again copy the values from the registers $n+1, n+2, \ldots, 2n$ to the registers $2n+1, 2n+2, \ldots, 3n$, and insert a modified version of N_2 where each register is shifted by $2n$, which ends with the value $\psi_2(\vec{x})$ in register $2n+1$. We then transfer this value to register 2, and zero out registers $3n+1$ through $\max\{3n+1, 2n+k\}$. Continue in this way m times until we have the values of each $\psi_i(\vec{x})$ in register i. Notice that if there exists an i with $\psi_i(\vec{x}) \uparrow$, then our machine will run forever, which agrees with the fact that θ outputs $\uparrow$ in this case.

Finally, we zero out registers $n+1$ through $3n$, and insert M with jump instructions shifted accordingly. The result of this computation will put the value $\theta(\vec{x}) = \varphi(\psi_1(\vec{x}), \psi_2(\vec{x}), \ldots, \psi_m(\vec{x}))$ into register 1 if $\varphi(\psi_1(\vec{x}), \psi_2(\vec{x}), \ldots, \psi_m(\vec{x})) \downarrow$, and will run forever otherwise, as desired.

- Closure under Primitive Recursion: Suppose that $\psi\colon \mathbb{N}^n \to \mathbb{N}^{\uparrow}$ and $\varphi\colon \mathbb{N}^{n+2} \to \mathbb{N}^{\uparrow}$ are both URM-computable. We show that $\theta = PrimRec(\psi, \varphi)$ is URM-computable. Fix a URM M that computes φ and a URM N that computes ψ. Let k be the maximum number of any register referred to in either of these machines. We describe a URM that computes θ.

  We first outline the overall idea. Recall that we can intuitively compute $\theta(\vec{x}, y)$ by first computing $\theta(\vec{x}, 0) = \psi(\vec{x})$, and then iterating through a loop a total of y times, each pass of which computes $\theta(\vec{x}, i)$ for increasing values of i. Since we will need them in each iteration of the loop, we leave the value of the input registers $1, 2, \ldots, n, n+1$ alone, until we have computed the correct final value, at which point we transfer it to register 1. We use register $n+2$ as a counter which will keep track of the number of times we have passed through the loop, so we will stop when the value in register $n+2$ equals the value in register $n+1$, which will be fixed at y throughout the computation. Finally, we will use register $n+3$ to hold the value $\theta(\vec{x}, i)$ after i iterations of the loop have been completed.

  We now describe the machine in more detail. On input $(\vec{x}, y) \in \mathbb{N}^{n+1}$, we copy the values of the registers $1, 2, \ldots, n$ to the registers $n+3, n+4, \ldots, 2n+2$. Next we insert a modified version of N where each register is shifted by $n+2$, which ends with the value $\psi(\vec{x}) = \theta(\vec{x}, 0)$ in register $n+3$. Since $r_{n+2} = 0$ at this point, notice that $r_{n+3} = \theta(\vec{x}, r_{n+2})$.

  We now begin a loop, assuming at the beginning that $r_{n+3} = \theta(\vec{x}, r_{n+2})$. First, check whether $r_{n+1} = r_{n+2}$, and if so, transfer the contents of register $n+3$ to register 1 and stop. We then copy the value of register $n+3$ to register $2n+4$, copy the value of register $n+2$ to register $2n+3$, copy the the values of the registers $1, 2, \ldots, n$ to the registers $n+3, n+4, \ldots, 2n+2$, and then zero out registers $2n+5$ through $\max\{n+2+k, 2n+5\}$. At this point, the contents of registers $n+3, n+4, \ldots, 2n+4$ are the values $\vec{x}, r_{n+2}, \theta(\vec{x}, r_{n+2})$, and they are followed by all zeros. Now we insert a modified version of M where each register is shifted by $n+2$, which ends with the value $\varphi(\vec{x}, r_{n+2}, \theta(\vec{x}, r_{n+2})) = \theta(\vec{x}, r_{n+2}+1)$ in register $n+3$. Next, increment the value in register $n+2$, and note that this establishes the invariant $r_{n+3} = \theta(\vec{x}, r_{n+2})$, but now we have increased the value r_{n+2} by 1. At this point, we jump to the beginning of the loop.

  We note that on input $(\vec{x}, y)$, we pass through the full loop a total of y times, incrementing the value of r_{n+2} once on each pass until it equals the value y. Then we finish by transferring the correct value $\theta(\vec{x}, y)$ to register 1. Finally, notice that if either $\psi(\vec{x}) \uparrow$ or $\theta(\vec{x}, r_{n+2}) \uparrow$ for any value that appears in r_{n+2}, then our machine runs forever, which agrees with the fact that θ outputs $\uparrow$ in these cases.

- Closure under Minimization: Suppose that $\psi\colon \mathbb{N}^{n+1} \to \mathbb{N}^{\uparrow}$ is URM-computable. We show that $\theta = Minimize(\psi)$ is URM-computable. Fix a URM M that computes ψ, and let k be the maximum number of any register referred to in M. We describe a URM that computes θ.

  We first outline the overall idea. Recall that we can intuitively compute $\theta(\vec{x})$ by first computing $\psi(\vec{x}, 0)$ and checking whether the result is 0, then computing $\psi(\vec{x}, 1)$ and checking whether the result is 0, etc.

We do this by creating a simple loop. Since we will need them in each iteration of the loop, we leave the value of the input registers $1, 2, \ldots, n$ alone, until we have computed the correct final value, at which point we transfer it to register 1. We will use register $n+1$ as the counter that will represent the last input to ψ, and we will leave register $n+2$ to hold the value 0 so that we can check the values of ψ against it.

We now describe the machine in more detail. On input $\vec{x} \in \mathbb{N}^n$, we begin a loop. We first copy the values of the registers $1, 2, \ldots, n+1$ to the registers $n+3, n+4, \ldots, 2n+3$. Next we insert a modified version of M where each register is shifted by $n+2$, which ends with the value $\psi(\vec{x}, r_{n+1})$ in register $n+3$. We then check whether $r_{n+2} = r_{n+3}$, and if so, we transfer r_{n+1} to register 1 and stop. Otherwise, we increment register $n+1$, and jump to the beginning of the loop.

It is now straightforward to check that our machine computes the correct value of ψ in all cases, including the situations where ψ outputs $\uparrow$ before it outputs 0, and where ψ never outputs 0.

□

Perhaps surprisingly, the converse of Theorem 12.4.4 is also true. In other words, despite the large differences in definitions and conceptual models, the class of partial recursive recursive functions is the same as the class of URM-computable functions. In order to prove this result, we start with a coding of URM machines by numbers. The essential point is that we can create such a coding where all of the important functions and relations are partial recursive (in fact, almost all of them will be primitive recursive). At a high level, this coding is similar to the coding of expressions that built up from the initial functions using *Compose* and *PrimRec* that we used before. However, since URM machines are not defined recursively, we can get by with a much simpler scheme than the one we used there.

Definition 12.4.5. *We code URM instructions as follows:*

1. *We use the number* $\langle 0, i \rangle = 2 \cdot 3^{i+1}$ *as the code for the instruction* $Z(i)$.
2. *We use the number* $\langle 1, i \rangle = 4 \cdot 5^{i+1}$ *as the code for the instruction* $S(i)$.
3. *We use the number* $\langle 2, i, j \rangle = 8 \cdot 3^{i+1} \cdot 5^{j+1}$ *as the code for the instruction* $T(i, j)$.
4. *We use the number* $\langle 3, i, j, q \rangle = 16 \cdot 3^{i+1} \cdot 5^{j+1} \cdot 7^{q+1}$ *as the code for the instruction* $J(i, j, q)$.

We let $InstrCode \subseteq \mathbb{N}$ *be the set of all valid instruction codes.*

Using these code numbers for instructions, we can now code each URM.

Definition 12.4.6. *Let* $M = (I_1, I_2, \ldots, I_s)$ *be a URM. If* $k_j \in \mathbb{N}$ *is the code of instruction* I_j*, then we use the number* $\langle k_1, k_2, \ldots, k_s \rangle$ *as the code for* M*. We let* $URMCode \subseteq \mathbb{N}$ *be the set of all codes of URM machines.*

For example, if $M = (J(2, 3, 5), S(1), S(3), J(1, 1, 1))$ is the URM we studied above with the property that $\varphi_M^{(2)}$ is the addition function, then the code of M is the number

$$\langle \langle 3, 2, 3, 5 \rangle, \langle 1, 1 \rangle, \langle 1, 3 \rangle, \langle 3, 1, 1, 1 \rangle \rangle.$$

The first basic fact that we need is the following.

Proposition 12.4.7. *The sets* $InstrCode$ *and* $URMCode$ *are both primitive recursive.*

Proof. We start with $InstrCode$. Notice that the set $R_0 = \{\langle 0, i\rangle : i \in \mathbb{N}^+\}$ is primitive recursive because

$$z \in R_0 \Leftrightarrow z \in Seq \wedge Len(z) = 2 \wedge z[0] = 0 \wedge z[1] > 0.$$

Similarly, each of the following sets is primitive recursive:

$$R_1 = \{\langle 1, i\rangle : i \in \mathbb{N}^+\}$$
$$R_2 = \{\langle 2, i, j\rangle : i, j \in \mathbb{N}^+\}$$
$$R_3 = \{\langle 3, i, j, q\rangle : i, j, q \in \mathbb{N}^+\}.$$

Since $InstrCode = R_0 \cup R_1 \cup R_2 \cup R_3$ and the primitive recursive relations are closed under union, it follows that $InstrCode$ is primitive recursive.

Next notice that

$$z \in URMCode \Leftrightarrow z \in Seq \wedge (\forall i < Len(z))(z[i] \in InstrCode).$$

Therefore, $URMCode$ is primitive recursive. □

We leave the following useful fact as an exercise.

Proposition 12.4.8. *Define a function $MaxRegister\colon \mathbb{N} \to \mathbb{N}$ by letting $MaxRegister(e)$ be the largest register number referred to in the machine coded by e when $e \in URMCode$, and letting $MaxRegister(e) = 0$ otherwise. The function $MaxRegister$ is primitive recursive.*

Proof. Exercise. □

With that setup in place, the interesting part of the converse to Theorem 12.4.4 is learning how to "understand" the dynamic execution of a URM in a primitive recursive way. The challenge here is that our mental model of the computation of a URM is that the state evolves over time, but primitive recursive functions have no direct analogue of state. In order to figure out how to resolve this issue, we return to our original concrete example of the URM $M = (J(2, 3, 5), S(1), S(3), J(1, 1, 1))$ executing from the initial state $1, 2, 0, 0, \ldots$:

$$\begin{aligned}(1) : 1, 2, 0, 0, \ldots &\mapsto (2) : 1, 2, 0, 0, \ldots\\ &\mapsto (3) : 2, 2, 0, 0, \ldots\\ &\mapsto (4) : 2, 2, 1, 0, \ldots\\ &\mapsto (1) : 2, 2, 1, 0, \ldots\\ &\mapsto (2) : 2, 2, 1, 0, \ldots\\ &\mapsto (3) : 3, 2, 1, 0, \ldots\\ &\mapsto (4) : 3, 2, 2, 0, \ldots\\ &\mapsto (1) : 3, 2, 2, 0, \ldots\\ &\mapsto (5) : 3, 2, 2, 0, \ldots\end{aligned}$$

Although each line of this computation consists of an infinite sequence, we do not need to carry around the trailing zeros that begin at register 4. In general, recall that a URM refers to only finitely many registers, so if the initial state consists of only finitely many nonzero registers, then there are only finitely many registers that will have a nonzero value at any point during the machine's execution. Thus, we can code each of the above stages of computation with a finite sequence, which we can then code as a number. For example, we can code the initial state as either the number $\langle 1, 1, 2\rangle$ or the number $\langle 1, 1, 2, 0\rangle$, where we put the instruction number in position 0 and assume that everything after the sequence ends is a 0. In other words, we are

taking one of the infinite sequences f_m that we used in the formal definition of $\varphi_M^{(n)}$, cutting off the trailing zeros at some point, and coding the resulting finite sequence as a number.

We call an element of Seq that is interpreted in this way (i.e. the first element is the next instruction number to be executed, and the remaining numbers are the values of the registers in order, with the assumption that all other registers have value 0) a *configuration* of a machine at a stage of computation. Notice that every element of Seq with positive length whose first element is positive can be interpreted as a configuration, which leads to the following definition.

Definition 12.4.9. *Let*

$$Config = \{c \in \mathbb{N} : c \in Seq, Len(c) > 0, \text{ and } c[0] > 0\}.$$

Since Seq, Len, and $DecodeSeq$ are all primitive recursive, we immediately obtain the following.

Proposition 12.4.10. *The set $Config$ is primitive recursive.*

Given a URM M together with the next instruction to execute and the current values in the registers, we can easily determine what happens after one step of execution. Essentially, we are saying that the definition of f_{m+1} from f_m is "simple". To make this more precise, we now code everything in sight as numbers so that we can argue that the corresponding relation is primitive recursive.

Definition 12.4.11. *We define a relation $StepCompute \subseteq \mathbb{N}^3$ by letting $(e, c, d) \in StepCompute$ if and only if all of the following are true:*

1. *$e \in URMCode$ and $c, d \in Config$.*
2. *$Len(c) = Len(d)$.*
3. *$Len(c) > MaxRegister(e)$.*
4. *If we let M be the machine coded by e, let f be the infinite sequence obtained by taking the finite sequence coded by c and appending an infinite sequence of zeros to the end, and let g be the corresponding infinite sequence coded by d, then g is obtained from f by one step of computation of M as described just before Definition 12.4.2.*

The first requirement is self-explanatory. Although the second requirement is not strictly necessary, this assumption allows us to "see" the same registers in the two configurations, which will simplify a few of the technical arguments below. Now the third requirement says that the configurations c and d (the latter because it has the same length as c) are long enough that any action of the machine coded by e will only affect registers coded by these configurations. Finally, the fourth requirement encapsulates the essential condition that we are after.

As an example, if

$$e = \langle\langle 3, 2, 3, 5\rangle, \langle 1, 1\rangle, \langle 1, 3\rangle, \langle 3, 1, 1, 1\rangle\rangle$$

is the code of our addition machine, then

$$(e, \langle 1, 1, 2, 0\rangle, \langle 2, 1, 2, 0\rangle) \in StepCompute$$

and

$$(e, \langle 3, 3, 2, 1\rangle, \langle 4, 3, 2, 2\rangle) \in StepCompute.$$

According to our definition, we have

$$(e, \langle 1, 1, 2\rangle, \langle 2, 1, 2\rangle) \notin StepCompute$$

because $MaxRegister(e) = Len(\langle 1,1,2\rangle) = 3$, so $\langle 1,1,2\rangle$ is not long enough to include all of the registers referred to in our machine. In contrast, we can add extra zeros to the end of the configuration codes, so we do have

$$(e, \langle 1,1,2,0,0\rangle, \langle 2,1,2,0,0\rangle) \in StepCompute.$$

The next result is the most important step on the way to proving the converse to Theorem 12.4.4.

Proposition 12.4.12. *The relation $StepCompute \subseteq \mathbb{N}^3$ is primitive recursive.*

Proof. Let $(e,c,d) \in \mathbb{N}^3$ be arbitrary. We need to express the statement $(e,c,d) \in StepCompute$ in a primitive recursive way. Notice that the first three conditions in the definition of $StepCompute$ can be expressed as saying

$$e \in URMCode \wedge c \in Config \wedge d \in Config \wedge Len(c) = Len(d) \wedge Len(c) > MaxRegister(e).$$

We will now proceed to work out the details of how to express the fourth condition in a primitive recursive way. The basic idea is that we look at $c[0]$, which is the number of the next instruction to execute. Now if $c[0] \geq Len(e)$, then there is no next instruction, so we should have $c = d$. We can express this as

$$c[0] \geq Len(e) \to c = d.$$

Otherwise, $c[0]$ corresponds to an instruction of the machine coded by e. To determine that instruction, we look at $e[c[0]]$ to pull out the corresponding instruction code, and then look at the value in the first position of this sequence. In other words, we have to examine the value $e[c[0]][0]$. If this number is 0, then we know that the next instruction is of type $Z(i)$, and we can determine i by looking at the value $e[c[0]][1]$. With all of this background in hand, we need to check that $d[0] = c[0]+1$ (i.e. that we move on to the next instruction in order), that we zero out the correct register in d, and that we leave all other registers alone. Putting everything together, we can express this as follows:

$$\begin{aligned}(c[0] < Len(e) \wedge e[c[0]][0] = 0) \to (&d[0] = c[0] + 1 \wedge d[e[c[0]][1]] = 0 \\ &\wedge (\forall i < Len(c))((i > 0 \wedge i \neq e[c[0]][1]) \to d[i] = c[i])).\end{aligned}$$

Now if instead $e[c[0]][0] = 1$, then we know that the next instruction is of type $S(i)$. Following the above logic, we can obtain

$$\begin{aligned}(c[0] < Len(e) \wedge e[c[0]][0] = 1) \to (&d[0] = c[0] + 1 \wedge d[e[c[0]][1]] = c[e[c[0]][1]] + 1 \\ &\wedge (\forall i < Len(c))((i > 0 \wedge i \neq e[c[0]][1]) \to d[i] = c[i])).\end{aligned}$$

Moving on to the transfer instruction, we obtain

$$\begin{aligned}(c[0] < Len(e) \wedge e[c[0]][0] = 2) \to (&d[0] = c[0] + 1 \wedge d[e[c[0]][2]] = c[e[c[0]][1]] + 1 \\ &\wedge (\forall i < Len(c))((i > 0 \wedge i \neq e[c[0]][2]) \to d[i] = c[i])).\end{aligned}$$

Finally, we need to handle the jump instructions. In this situation, we have two separate cases. Working out the details, we end up at

$$\begin{aligned}(c[0] < Len(e) \wedge e[c[0]][0] = 3 \wedge c[e[c[0]][1]] = c[e[c[0]][2]]) \\ \to (d[0] = q \wedge (\forall i < Len(c))(0 < i \to d[i] = c[i]))\end{aligned}$$

and

$$\begin{aligned}(c[0] < Len(e) \wedge e[c[0]][0] = 3 \wedge c[e[c[0]][1]] \neq c[e[c[0]][2]]) \\ \to (d[0] = c[0] + 1 \wedge (\forall i < Len(c))(0 < i \to d[i] = c[i])).\end{aligned}$$

Putting everything together, we have that $(e, c, d) \in StepCompute$ if and only if (e, c, d) satisfies the conjunction of all of the displayed lines above. Since Len and $DecodeSeq$ are both primitive recursive, and the primitive recursive relations are closed under Boolean operations and primitive recursively bounded quantification, it follows that $StepCompute$ is primitive recursive. □

We now use $StepCompute$ to definition one more fundamental relation $T \subseteq \mathbb{N}^3$. Although the formal definition looks elaborate, the underlying idea is elegant and straightforward. Intuitively, $(e, z, p) \in T$ if and only if p codes a complete computation (i.e. a computation that eventually stops) via a sequence of configurations of the machine coded by e on initial input z.

Definition 12.4.13. *We define a relation $T \subseteq \mathbb{N}^3$ by letting $(e, z, p) \in T$ if all of the following are true:*

1. *$e \in URMCode$, $z \in Seq$, and $p \in Seq$.*
2. *$Len(p) > 0$.*
3. *$StepCompute(e, p[i], p[i+1])$ for all $i < Len(p) - 1$.*
4. *$Len(p[0]) > Len(z)$.*
5. *$p[0][0] = 1$, and for all $i < Len(z)$, we have $z[i] = p[0][i+1]$.*
6. *For all $i < Len(p[0])$, if $i \geq Len(z)$, then $p[0][i] = 0$.*
7. *$p[Len(p) - 1][0] > Len(e)$.*

The relation T is known as Kleene's T-predicate.

Again, the first requirement is self-explanatory. The second and third requirements say that p is a nonempty sequence of configurations, each of which follows from the previous one via exactly one step of the action of the machine coded by e. Now the next three requirements simply say that the configuration $p[0]$ begins with 1 (the first instruction to execute), and that the register values of $p[0]$ correspond with the entries in the input z, modulo some additional zeros that might be on the end of $p[0]$. Notice that the index we use for z is one less than the index we use for $p[0]$ in the fifth line because the first element of $p[0]$ corresponds to an instruction number, but z has no such entry. The final requirement says that the instruction number in the last configuration of p does not correspond to a valid instruction of e, which means that the computation has come to an end.

Proposition 12.4.14. *Kleene's T-predicate is primitive recursive.*

Proof. Immediate from the fact that $URMCode$, Seq, $DecodeSeq$, Len, and $StepCompute$ are all primitive recursive. □

It will also be useful to have a small variant of the T-predicate, where instead of using an element of Seq to code the input to the machine, we simply use an actual sequence of numbers.

Definition 12.4.15. *For each $n \in \mathbb{N}^+$, define a relation $T_n \subseteq \mathbb{N}^{n+2}$ by*

$$T_n = \{(e, x_1, x_2, \ldots, x_n, p) \in \mathbb{N}^{n+2} : (e, \langle x_1, x_2, \ldots, x_n \rangle, p) \in T\}.$$

Since T is primitive recursive, we immediately obtain the following.

Proposition 12.4.16. *T_n is primitive recursive for each $n \in \mathbb{N}^+$.*

Suppose that $(e, z, p) \in T$. In this case, the number e is a code for a machine M acting on the input coded by z, and p codes a sequence of configurations that corresponding to a complete computation. Can we determine the final output of that computation? Recall that once a URM has completed computation, we define the output to be the value in the first register. Thus, we need only examine the value $p[Len(p) - 1][1]$ to know the output of the computation, which motivates the following definition.

Definition 12.4.17. *Define a function $U : \mathbb{N} \to \mathbb{N}$ by letting*

$$U(p) = \begin{cases} p[Len(p) - 1][1] & \text{if } p \in Seq, Len(p) > 0, p[Len(p) - 1] \in Seq, \text{ and } Len(p[Len(p) - 1]) > 2 \\ 0 & \text{otherwise.} \end{cases}$$

Proposition 12.4.18. *The function U is primitive recursive.*

By combining Kleene's T-predicate with the function U, we can easily express whether a computation stops at some stage, and if so, extract the corresponding output of the computation. In order to connect them, we simply perform a (potentially unbounded) search for a number that codes a successful computation.

Proposition 12.4.19. *Let M be a URM and let $n \in \mathbb{N}^+$. Let $e \in \mathbb{N}$ be the code of M.*

1. *For all $\vec{x} \in \mathbb{N}^n$, we have $\varphi_M^{(n)}(\vec{x}) \downarrow$ if and only if $\exists p\ T_n(e, \vec{x}, p)$.*
2. *For all $\vec{x} \in \mathbb{N}^n$ and all $p \in \mathbb{N}$ such that $T_n(e, \vec{x}, p)$, we have $\varphi_M^{(n)}(\vec{x}) = U(p)$.*

In particular, we have $\varphi_M^{(n)}(\vec{x}) = U(\mu p\ T_n(e, \vec{x}, p))$ for all $\vec{x} \in \mathbb{N}^n$, assuming that we interpret $U(\uparrow)$ to be $\uparrow$ (as we did when we defined Compose on partial functions).

Proof. Immediate. □

With all of that hard work in hand, we can now easily prove the converse of Theorem 12.4.4.

Theorem 12.4.20. *Every URM-computable partial function is partial recursive.*

Proof. Let $\psi : \mathbb{N}^n \to \mathbb{N}^{\uparrow}$ be a partial recursive function. Fix a URM M that computes ψ, i.e. such that $\psi = \varphi_M^{(n)}$, and let e be the code of M. By Proposition 12.4.19, we have

$$\psi(\vec{x}) = U(\mu p\ T_n(e, \vec{x}, p))$$

for all $\vec{x} \in \mathbb{N}^n$. Since T_n is a primitive recursive relation, it is a recursive relation, so we can use Proposition 12.3.15 to conclude that the partial function $(e, \vec{x}) \mapsto \mu p\ T_n(e, \vec{x}, p)$ is partial recursive. Since U is primitive recursive, it is partial recursive, and hence $\psi(\vec{x}) = U(\mu p\ T_n(e, \vec{x}, p))$ is partial recursive. □

We can also use Proposition 12.4.19 to prove the existence of a *universal machine*. That is, there exists a single URM N that can simulate any other URM when provided with its code number. This result is not too surprising from a modern programming perspective, as one can write one program that acts as an interpreter for all programs. Nonetheless, it was historically important (when first proved by Turing in the context of Turing machines), and it will play a fundamental role in later arguments. We state the result in two forms corresponding to our two versions of the T-predicate.

Proposition 12.4.21.

1. *There exists a URM N such that*

$$\varphi_N^{(2)}(e, z) = \begin{cases} \varphi_M^{(n)}(x_1, \ldots, x_n) & \text{if } e \in URMCode \text{ is the code of } M \text{ and } z = \langle x_1, \ldots, x_n \rangle \in Seq \\ 0 & \text{otherwise.} \end{cases}$$

In other words, $\varphi_N^{(2)}(e, z)$ is the result of running the URM coded by e on the sequence coded by z.

2. Let $n \in \mathbb{N}^+$. There exists a URM N such that

$$\varphi_N^{(n+1)}(e, \vec{x}) = \begin{cases} \varphi_M^{(n)}(\vec{x}) & \text{if } e \in URMCode \text{ is the code of } M \\ 0 & \text{otherwise.} \end{cases}$$

In other words, $\varphi_N^{(n+1)}(e, \vec{x})$ is the result of running the URM coded by e on the sequence $\vec{x}$.

Proof. Define a partial function $\psi \colon \mathbb{N}^2 \to \mathbb{N}^{\uparrow}$ by

$$\psi(e, z) = \begin{cases} U(\mu p\ T(e, z, p)) & \text{if } e \in URMCode \text{ and } z \in Seq \\ 0 & \text{otherwise.} \end{cases}$$

Since T, $URMCode$, and Seq are all recursive relations, and U is a partial recursive function, we can use Proposition 12.3.14 to conclude that ψ is a partial recursive function. Using Theorem 12.4.4, it follows that ψ is URM-computable, so we can fix a URM N with $\varphi_N^{(2)} = \psi$, which gives the first result. For the second, apply the same argument to the partial function $\psi \colon \mathbb{N}^{n+1} \to \mathbb{N}^{\uparrow}$ defined by

$$\psi(e, \vec{x}) = \begin{cases} U(\mu p\ T_n(e, \vec{x}, p)) & \text{if } e \in URMCode \\ 0 & \text{otherwise.} \end{cases}$$

□

The existence of a universal machine allows us to draw a crucial distinction between the class of partial recursive functions and the class of primitive recursive functions. Since the class of partial recursive functions is the same as the class of URM-computable functions, we obtain the following consequence part 2 of Proposition 12.4.21 in the case where $n = 1$: There exists a partial recursive function $\psi \colon \mathbb{N}^2 \to \mathbb{N}^{\uparrow}$ such that every unary partial recursive function appears as a "row" of ψ. That is, there exists a partial recursive function $\psi \colon \mathbb{N}^2 \to \mathbb{N}^{\uparrow}$ such that for all partial recursive functions $\varphi \colon \mathbb{N} \to \mathbb{N}^{\uparrow}$, there exists $e \in \mathbb{N}$ such that $\psi(e, x) = \varphi(x)$ for all $x \in \mathbb{N}$.

Is the corresponding result true for primitive recursive functions? In other words, does there exist a primitive recursive function $g \colon \mathbb{N}^2 \to \mathbb{N}$ such that for all primitive recursive functions $f \colon \mathbb{N} \to \mathbb{N}$, there exists $e \in \mathbb{N}$ such that $g(e, x) = f(x)$ for all $x \in \mathbb{N}$? That answer is no, because the existence of such a g would allow us to diagonalize out of the primitive recursive functions in a primitive recursive way! In other words, suppose that such a g existed. Define $f \colon \mathbb{N} \to \mathbb{N}$ by letting $f(e) = g(e, e) + 1$ for all $e \in \mathbb{N}$, and notice that f is primitive recursive. Thus, by our assumed property of g, we can fix an e such that $g(e, x) = f(x)$ for all $x \in \mathbb{N}$. We then have that $f(e) = g(e, e) + 1 = f(e) + 1$, which is a contradiction.

Notice that this argument closely mirrors the argument that we gave at the end of Section 12.2. In that setting, we established a coding for expressions that were built up from the functions in $Init$ using finitely many applications of $Compose$ and $PrimRec$. That is, we coded each such expression with a number $e \in PrimRecCode$, just like how we have now coded each URM with a number $e \in URMCode$. We also recursively defined a function Φ that takes a number $e \in PrimRecCode$ and produces the primitive recursive function $\Phi(e)$ that it codes. The table at the end of that section was the function $h \colon \mathbb{N}^2 \to \mathbb{N}$ defined by

$$h(e, x) = \begin{cases} \Phi(e)(x) & \text{if } e \in PrimRecCode_1 \\ 0 & \text{otherwise,} \end{cases}$$

i.e. $h(e, x)$ is the output of the primitive recursive function coded by e on input x. We then argued that the function obtained by taking the diagonal of h and adding 1 to the result was not primitive recursive. In light of the previous paragraph, we can also say that h itself is not primitive recursive, because otherwise the corresponding function $f(e) = h(e, e) + 1$ would be primitive recursive.

Although h is not primitive recursive, we did argue that it was "intuitively computable". With the existence of a universal machine, we can now outline an argument that h is a total recursive function, from which it follows that the diagonalization $f(e) = h(e, e) + 1$ is an example of a total recursive unary function that is not primitive recursive. We begin by noticing that the proof of Theorem 12.4.4 is completely constructive. That is, given a URM M that computes $\varphi\colon \mathbb{N}^m \to \mathbb{N}^\uparrow$ together with URMs $N_1, N_2, \ldots, N_m$ that compute $\psi_1, \psi_2, \ldots, \psi_m\colon \mathbb{N}^n \to \mathbb{N}^\uparrow$, we explicitly built a URM that computes $Compose(\varphi, \psi_1, \psi_2, \ldots, \psi_m)$. More formally, it can be shown through a careful detailed analysis of this argument that there exists a primitive recursive function $g\colon \mathbb{N} \to \mathbb{N}$ such that if e is a code for a URM that computes φ, and i_k is a code for a URM that computes ψ_k, then $g(\langle e, i_1, i_2, \ldots, i_m\rangle)$ is a code for a URM that computes $Compose(\varphi, \psi_1, \psi_2, \ldots, \psi_m)$. The argument for $PrimRec$ is similarly constructive (as is the argument for $Minimize$, but that does not concern us here). Using these results together with order recursion, it follows that there exists a primitive recursive function $\alpha\colon \mathbb{N} \to \mathbb{N}$ such that if $e \in PrimRecCode_n$, then $\alpha(e) \in URMCode$, and $\Phi(e) = \varphi_M^{(n)}$, where M is the URM coded by $\alpha(e)$. That is, we can primitive recursively turn an element of $PrimRecCode_n$ into an element of $URMCode$ in such a way that the underlying functions are identical. Now let N be a URM that is guaranteed to exist by part 2 of Proposition 12.4.21 in the case where $n = 1$. We then have

$$h(e, x) = \begin{cases} \varphi_N^{(2)}(\alpha(e), x) & \text{if } e \in PrimRecCode_1 \\ 0 & \text{otherwise.} \end{cases}$$

Since $PrimRecCode_1$ and α are both primitive recursive, and $\varphi_N^{(2)}$ is partial recursive, it follows that h is partial recursive. Moreover, since $\Phi(e)$ is a total function for each $e \in PrimRecCode_1$, we conclude that h is total.

Before moving on to discuss whether the class of partial recursive functions (which is the same as the class of URM-computable functions) actually encompasses all "intuitively computable" functions, we first establish a couple of surprising consequences of all of our hard work in this section. The first is that although we can use $Minimize$ several times in order to show that a given partial function is partial recursive, we never actually need to use $Minimize$ more than once for any given partial function. In other words, if we think of $Minimize$ as our analogue of a while-loop in a programming language (while thinking of $PrimRec$ as our analogue of a for-loop), then every program can be written using at most one while-loop. This interesting fact follows immediately from Proposition 12.4.19 because T_n and U are both primitive recursive total functions, and hence we do not need to use $Minimize$ at all for them.

For the second, recall that when we first decided to move on from primitive recursive functions, we argued that we should embrace partial functions. We also noted that a machine model of computation like our URM model naturally leads to partial functions since a machine might run forever on some inputs. However, we can now argue that it is possible to generate the class of all total recursive functions without referring to partial functions at any point. The key is to restrict applications of $Minimize$ (the place where partial functions first arise) to situations where the the resulting function is total, i.e. when the function g has the following property.

Definition 12.4.22. *Let $n \in \mathbb{N}^+$, and let $g\colon \mathbb{N}^{n+1} \to \mathbb{N}$ be a total function. We say that g is* regular *if for every $\vec{x} \in \mathbb{N}^n$, there exists $y \in \mathbb{N}$ with $g(\vec{x}, y) = 0$.*

Although it is easy to see that every function generated under this restriction is a total recursive function, it is not at all obvious that we obtain all total recursive functions in this way. Surprisingly, we establish this fact by using our hard work on URMs that culminated in Proposition 12.4.19.

Proposition 12.4.23. *The collection of all total recursive functions is the subset of all total functions obtained by starting with $Init$, and generating using $Compose$, $PrimRec$, and a restricted version of $Minimize$ where we can only form $Minimize(g)$ in the case where g is a regular function.*

Proof. Let $\mathcal{G}$ be the collection that is generated by starting with the $Init$, and generating using $Compose$, $PrimRec$, and our restricted version of $Minimize$. Certainly, every function in $\mathcal{G}$ is partial recursive. Also, note that every primitive recursive function is in $\mathcal{G}$ trivially.

Since our restricted version of $Minimize$ outputs only total functions on total inputs, and the same is true for $Compose$ and $PrimRec$, it follows that every function in $\mathcal{G}$ is total. Therefore, every function in $\mathcal{G}$ is a total recursive function.

Conversely, let $h\colon \mathbb{N}^n \to \mathbb{N}$ be a total recursive function. We then have that h is URM-computable by Theorem 12.4.4. Fix a URM M that computes h, i.e. such that $h = \varphi_M^{(n)}$, and let e be the code of M. Since $\varphi_M^{(n)}(\vec{x}) \downarrow$ for all $\vec{x} \in \mathbb{N}^n$, we know that for every $\vec{x} \in \mathbb{N}^n$, there exists $p \in \mathbb{N}$ with $T_n(e, \vec{x}, p)$. Define $g\colon \mathbb{N}^{n+1} \to \mathbb{N}$ by letting $g(\vec{x}, p) = 1 \dot{-} \chi_{T_n}(e, \vec{x}, p)$, so $g(\vec{x}, p) = 0$ if and only if $T_n(e, \vec{x}, p)$. Since T_n and $\dot{-}$ are both primitive recursive, we know that g is primitive recursive, so $g \in \mathcal{G}$. Furthermore, for every $\vec{x} \in \mathbb{N}^n$, there exists $p \in \mathbb{N}$ with $g(\vec{x}, p) = 0$, so g is regular. Thus, we can use our restricted version of $Minimize$ to conclude that the function $f = Minimize(g)$ is an element of $\mathcal{G}$. Now $h = Compose(U, f)$ by Proposition 12.4.20, so as $U \in \mathcal{G}$ because U is primitive recursive, it follows that $h \in \mathcal{G}$. □

In fact, this proof combines with the above discussion to establish a stronger result: Every total recursive function can be obtained from the functions in $Init$ by using finitely many applications of $Compose$ and $PrimRec$, and at most one application of the restricted version of $Minimize$.

12.5 Computability and the Church-Turing Thesis

We have introduced three different attempts to capture the notion of an "intuitively computable" function from $\mathbb{N}^n$ to $\mathbb{N}$ in this chapter. We began with the primitive recursive functions. Although we established that this class was broad enough to encompass the vast majority of the natural functions that arise in practice, we did construct a somewhat unnatural function via diagonalization that was not primitive recursive, but that we could certainly compute. In an attempt to remedy this situation, we then introduced two class of partial functions: partial recursive functions and URM-computable functions. Although they came from different perspectives, we eventually argued that these two classes coincide. Moreover, because we switched to partial functions in these models, the diagonalization argument that showed the limitations of the primitive recursive functions does not directly apply (Proposition 12.4.23 might give us pause here, but we will discuss this interesting situation at the end of the section).

One might look at our definition of a URM, note that the instructions are incredibly basic, and wonder whether we could enlarge the class of partial functions that URMs can compute if we just include more instruction types. After all, there are many (many) more low level machine instructions in every modern computing chip. Typically, it is easy to see that a proposed new instruction can be simulated by a suitable sequence of our basic ones, so adding the new instruction does not enlarge the class of functions that can be computed. However, we do not need to think through the details of each such proposed instruction, because we can give a high level argument that whenever we consider adding a "reasonable" finite set of instructions to the definition of a URM, then every function that can be computed by these enhanced URMs will still be partial recursive.

To see this, consider the situation where we add a finite set of "reasonable" instructions to the definition of a URM. Assuming that each of these instructions only refer to finitely many registers (part of our "reasonable" condition), we can then code these instructions as numbers, and use those numbers to code our enhanced URMs as numbers. Now as long as these new instructions behave in a way that the description of how to determine f_{m+1} from f_m (from just before Definition 12.4.2) is still "simple", then the new analogue of $StepCompute$ will still be primitive recursive. As a result, the new analogue of Kleene's T-predicate will still be primitive recursive, and hence the proof of Theorem 12.4.20 and all subsequent results in the previous section will go through without alteration.

What if we consider other models of computation? For example, we mentioned Turing machines in the last section. For that model, we can follow the ideas outlined in the previous section to define a code for each Turing machine, and also to define a configuration by taking the finitely many symbols currently written on the tape, the position of the head, and the current state, and coding them all together into one number. Next, we can define an analogue of *StepCompute*, and argue that the result is still primitive recursive. From there, the corresponding analogue of Kleene's T-predicate is easily seen to be primitive recursive. In this situation we have to do a bit more coding in a few places, like for the U function (since the output of a Turing machine is a finite string of symbols from the finite alphabet, rather than a number), but everything goes through as before. Alternatively, it is possible to give direct simulation arguments, modulo suitable coding, that a partial function is URM-computable if and only if it is Turing machine computable.

What about other models of computation? Enhancement to the Turing machine model that involve multiple tapes, or an infinite multiple grid of cells, result in the same fate as enhancements to the URM-model. But other approaches have arisen. One of the earliest attempts was using the so-called *λ-calculus*. Pioneered by Alonzo Church, this approach closely matches the paradigm of modern functional programming languages like ML and Haskell. Turing, who developed his model independently at around the same time, proved that the Turing machine computable functions are precisely the same as the λ-definable functions in Church's model. Turing's argument appears in an appendix to his groundbreaking paper that introduced Turing machines, universal machines, the halting problem (see below), and the resolution of one of Hilbert's problems. Many other less well-known models, including a machine model due to Post, and an equation model pioneered by Herbrand and Gödel, also define the same class of functions. We will eventually even define a class of functions using first-order logic that coincides with the URM-computable functions.

All of this work shows that the class of URM-computable functions is incredibly robust. If we change the URM model by adding other reasonable instructions, we arrive at the same class. If we try completely different approaches, we arrive at the same class. Moreover, if we examine the functions that are computable by a modern computer, but idealize to the situation where we have an unlimited stock of memory, then it can be shown (essentially because all of the instructions satisfy the "reasonable" requirement) that these functions are precisely the URM-computable functions. With so much evidence, it starts to become natural to believe that URMs encompass all general purpose computation.

The statement that the URM-computable functions, or the functions from any of the equivalent models, coincide with the "intuitively computable" functions is known as the *Church-Turing Thesis*. Of course, since this latter class is not formally defined, such a statement can not be proven in any mathematical sense. Church originally proposed his λ-definable functions had this property, but the evidence at the time was lacking. When Turing introduced his definition (the first formal machine model of computation), he also gave a high level argument that anything that a human can compute by hand can also be computed by one of his machines. Taking together with Turing's proof that the Turing machine computable functions were the same as the λ-definable functions, other mathematicians like Gödel began to believe that the "intuitively computable" functions had been successfully defined. The last 80 years of progress in the subject have only solidified this view. We use this evidence as motivation to drop the prefix URM from our definitions.

Definition 12.5.1. *Let $n \in \mathbb{N}^+$ and let $\psi\colon \mathbb{N}^n \to \mathbb{N}^{\uparrow}$ be a partial function. We say that ψ is* partial computable*, or simply* computable*, if it URM-computable. If ψ say also total, then we say that ψ is a* total computable function*.*

As alluded to in the definition, since we are now used to working with partial functions, we will occasionally drop the adjective partial in the future. However, we will also sometimes continue to use the words partial and total for emphasis.

Definition 12.5.2. *Let $n \in \mathbb{N}^+$ and let $R \subseteq \mathbb{N}^n$. We say that R is* computable *if its characteristic function χ_R is a total computable function.*

Note that many sources use the word *recursive* rather than *computable* in these definitions. Historically, this comes from the definition of partial recursive (also called general recursive) functions. However, the

word recursive has many meanings in mathematics, so we choose to follow the modern practice of adopting the more evocative word computable.

Even though we will use the somewhat loaded word *computable* from now on, keep in mind that everything does go back to our two formal definitions: URM-computable and partial recursive. It is true that many sources will sometimes describe a vague intuitive algorithm that purportedly gives the same output as a given partial function ψ, and then use this to simply assert that ψ is computable, without using one of the formal definitions. Such an argument is sometimes called a "a proof by the Church-Turing thesis". Of course, this language is misleading and confusing to the uninitiated. But with enough experience, it becomes completely routine to turn any rough algorithm described in high level language or pseudocode into a careful proof that ψ is partial recursive. In fact, we will occasionally follow this practice when the details are completely routine.

Since we have introduced a way to code URMs with numbers, we now adopt a slightly modified notation for $\varphi_M^{(n)}$.

Definition 12.5.3. *Let $e \in \mathbb{N}$ and let $n \in \mathbb{N}^+$. We define a partial function $\varphi_e^{(n)} \colon \mathbb{N}^n \to \mathbb{N}^\uparrow$ as follows. If $e \in URMCode$, then we define $\varphi_e^{(n)}$ to be $\varphi_M^{(n)}$, where M is the URM with code number e. If $e \notin URMCode$, then we define $\varphi_e^{(n)}(\vec{x}) = 0$ for all $\vec{x} \in \mathbb{N}^n$.*

In the case of unary functions, we often drop the superscript.

Notation 12.5.4. *Let $e \in \mathbb{N}$. We use the notation φ_e in place of $\varphi_e^{(1)}$.*

Since we have chosen to define $\varphi_e^{(n)}$ to be a (very trivial) URM-computable function in the case where $e \notin URMCode$, we have the following result.

Proposition 12.5.5. *Let $n \in \mathbb{N}^+$ and let $\psi \colon \mathbb{N}^n \to \mathbb{N}^\uparrow$ be a partial function. We then have that ψ is computable if and only if there exists an $e \in \mathbb{N}$ with $\psi = \varphi_e^{(n)}$.*

This minor shift in notation has several advantages. First, it provides a canonical listing of the primitive recursive functions. Second, by dropping a direct reference to URMs, it more closely aligns with the fact that there are many other equivalent definitions for the class of partial computable functions. Moreover, it is possible to provide a coding in each of these other models (like we did for primitive recursive functions, and alluded to for partial recursive functions). Of course, given a particular ψ, a code number e for which $\psi = \varphi_e^{(n)}$ might vary from model to model, but this fact is not important. After all, there are many other possible codings of URMs. Finally, another advantage of our notational shift is that we can often state results with more compact notation. That is, instead of saying

$$\text{There exists a partial recursive function } \psi \colon \mathbb{N}^n \to \mathbb{N} \text{ such that } \ldots \psi \ldots,$$

or

$$\text{There exists a URM } M \text{ such that } \ldots \varphi_M^{(n)} \ldots",$$

we can now say

$$\text{There exists } e \in \mathbb{N} \text{ such that } \ldots \varphi_e^{(n)} \ldots".$$

For example, we can restate the first part of Proposition 12.4.21 on the existence of a universal machine by saying that there exists an $i \in \mathbb{N}$ such that

$$\varphi_i^{(2)}(e, \langle x_1, x_2, \ldots, x_n \rangle) = \varphi_e^{(n)}(x_1, x_2, \ldots, x_n)$$

for all $e, x_1, x_2, \ldots, x_n \in \mathbb{N}$. Similarly, we can restate the second part as saying that for every $n \in \mathbb{N}^+$, there exists an $i \in \mathbb{N}$ such that

$$\varphi_i^{(n+1)}(e, \vec{x}) = \varphi_e^{(n)}(\vec{x})$$

for all $e \in \mathbb{N}$ and all $\vec{x} \in \mathbb{N}^+$. Here is another example.

Proposition 12.5.6. *For all $n \in \mathbb{N}^+$ and all $e \in \mathbb{N}$, there exists $i \in \mathbb{N}$ such that*

$$\varphi_i(\langle x_1, x_2, \ldots, x_n \rangle) = \varphi_e^{(n)}(x_1, x_2, \ldots, x_n)$$

for all $x_1, x_2, \ldots, x_n \in \mathbb{N}$.

In other words, every n-ary partial computable function can be simulated by a unary partial computable function via a sequence coding of the input.

Proof. Let $n \in \mathbb{N}^+$ and $e \in \mathbb{N}$ be arbitrary. Define $\psi \colon \mathbb{N} \to \mathbb{N}^{\uparrow}$ by

$$\psi(z) = \begin{cases} \varphi_e^{(n)}(z[0], z[1], \ldots, z[n-1]) & \text{if } z \in Seq \text{ and } Len(z) = n \\ 0 & \text{otherwise.} \end{cases}$$

Since $\varphi_e^{(n)}$ is computable, it is partial recursive, so ψ is partial recursive by Proposition 12.3.14. Since ψ is computable, we can fix an $i \in \mathbb{N}$ with $\varphi_i = \psi$. Note that φ_i has the required property. □

We have repeatedly stated that our move to partial functions impedes the diagonalization argument that doomed the primitive recursive functions. Let's examine that assertion more carefully. Consider the table where the entry in row e and column x is $\varphi_e(x)$:

$\varphi_e(x)$	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7
2	↑	↑	↑	↑	↑	↑	↑	↑
3	1	1	1	1	1	1	1	1
4	2	0	↑	0	0	0	0	0
5	1	↑	3	↑	5	↑	7	↑
6	0	1	1	2	3	5	8	13
7	8	0	7	15	↑	0	3	↑

Again, a large portion of the rows of this table should be filled with zeros in our coding (since many values of e are not valid codes of URMs), but we have chosen to fill in the entries with slightly more interesting values. One key observation is that the existence of a universal machine implies that this table is computable. More formally, we stated above that there exists an $i \in \mathbb{N}$ such that

$$\varphi_i^{(2)}(e, x) = \varphi_e(x)$$

for all $e \in \mathbb{N}$ and all $x \in \mathbb{N}$, so the partial function $(e, x) \mapsto \varphi_e(x)$ is computable. Since we can compute the table, why are we unable to computably diagonalize out? The first naive idea is to define $\psi \colon \mathbb{N} \to \mathbb{N}^{\uparrow}$ by letting $\psi(e) = \varphi_e(e) + 1$. Since $(e, x) \mapsto \varphi_e(x)$ is computable, the partial function ψ is computable. Therefore, ψ must appear as a row in the above table. The reason why this is not a contradiction is that there do exist values of e for which $\varphi_e(e) \uparrow$, like the values 2, 5, and 7 in the above table. For these values of e, we have $\psi(e) \uparrow$ since *Compose* produces $\uparrow$ whenever one of the inner functions does. Thus, all that we have shown is that whenever $e \in \mathbb{N}$ is a number with $\psi = \varphi_e$ (i.e. whenever ψ appears in row e), then we must have $\varphi_e(e) \uparrow$.

Can we get around the issue where some values of the diagonal equal $\uparrow$? One idea is to define a *total* function $f \colon \mathbb{N} \to \mathbb{N}$ by letting $f(e)$ be $\varphi_e(e) + 1$ whenever $\varphi_e(e) \downarrow$, and letting $f(e)$ be some value of $\mathbb{N}$, say 0, otherwise. It is then straightforward to check that f does not appear as any row of the table, so f is not computable. But what is underlying reason why f is not computable? After all, we simply defined f by cases, and in each case we used a known computable function (either $e \mapsto \varphi_e(e) + 1$ or $e \mapsto 0$). The only possible conclusion is that the case distinction is not computable. Working out the details, we arrive at the following fundamental result.

Theorem 12.5.7 (Turing)**.** *The set*

$$K = \{e \in \mathbb{N} : \varphi_e(e) \downarrow\}$$

is not computable.

Proof. Suppose instead that K was computable. Define a function $f\colon \mathbb{N} \to \mathbb{N}$ by letting

$$f(e) = \begin{cases} \varphi_e(e) + 1 & \text{if } e \in K \\ 0 & \text{otherwise.} \end{cases}$$

Since we are assuming that K is computable, and we know that $e \mapsto \varphi_e(e) + 1$ is computable (because $(e, x) \mapsto \varphi_e(x)$ is computable by the existence of a universal machine), it follows from Proposition 12.3.14 that f is computable. Thus, we can fix an i such that $f = \varphi_i$. Notice that $f(e) \downarrow$ for all $e \in \mathbb{N}$, so f is total. In particular, we have $f(i) \downarrow$, so $\varphi_i(i) \downarrow$. It follows that $\varphi_i(i) = f(i) = \varphi_i(i) + 1$, a contradiction. Therefore, K is not computable. □

In short, if K happened to be computable, then we would be able to computably diagonalize out of the computable functions, which is impossible. The question of whether K is computable is often called the *halting problem* because we are asking whether URMs eventually halt on certain inputs. However, there is a strange self-reference here, since K is the set of (codes of) URMs that eventually halt when the input is their own code. Instead of just looking at the diagonal, we can inquire about halting more generally. That is, it is more natural to to let the halting problem be the question of whether the set $\{(e, x) \in \mathbb{N}^2 : \varphi_e(x) \downarrow\}$ is computable.

Corollary 12.5.8. *The set*

$$K_0 = \{(e, x) \in \mathbb{N} : \varphi_e(x) \downarrow\}$$

is not computable.

Proof. If K_0 was computable, then since $e \in K \Leftrightarrow (e, e) \in K_0$, it would follow that K is computable. Since this would contradict Theorem 12.5.7, we conclude that that K_0 is not computable. □

Thus, both K and K_0 fail to be computable, and each one can be called the *halting set*. The proof of the corollary shows how we can "embed" the problem of determining K into the problem of determining K_0. It is also possible to show that we can "embed' the problem of determining K_0 into the problem of determining K. However, we will resist going down the road of formally defining what we mean by such an embedding, also known as a *reduction*, here. Nonetheless, there is a good reason why we often conflate the two sets.

Although K is not computable, there is a sense in which it is *almost* computable, and we will spend the remainder of this chapter exploring this idea. We begin by taking a slight detour to a modified version of Kleene's T-predicate. Recall the partial function which takes (e, z) as input, and produces the result of the computation of the URM coded by e operating on input z, is computable because it is $(e, z) \mapsto U(\mu p\ T(e, z, p))$. What if we instead take as input (e, z, s), and run the computation for exactly s steps, noting the resulting configuration at this stage? This function is certainly "intuitively computable", but we work out the details without just appealing to the Church-Turing Thesis. We use the generic letters R and f here because will only use them in passing as stepping stones.

Definition 12.5.9. *We define a relation $R \subseteq \mathbb{N}^4$ by letting $(e, z, s, p) \in R$ if (e, z, s, p) satisfies the first six requirements of Kleene's T-predicate, and also satisfies $Len(p) = s + 1$. More formally, $(e, z, s, p) \in R$ if all of the following are true:*

1. *$e \in URMCode$, $z \in Seq$, and $p \in Seq$.*
2. *$Len(p) > 0$.*

3. $StepCompute(e, p[i], p[i+1])$ *for all* $i < Len(p) - 1$.
4. $Len(p[0]) > Len(z)$.
5. $p[0][0] = 1$, *and for all* $i < Len(z)$, *we have* $z[i] = p[0][i+1]$.
6. *For all* $i < Len(p[0])$, *if* $i \geq Len(z)$, *then* $p[0][i] = 0$.
7. $Len(p) = s + 1$.

In other words, $(e, z, s, p) \in R$ if and only if $e \in URMCode$, $z \in Seq$, and p codes a computation that begins with the configuration consisting of instruction 1 and input coded by z, and continues from there for exactly s steps of computation. Notice that in contrast to Kleene's T-predicate, the computation might not be finished at this stage. However, like T, the set R is primitive recursive because all of the functions mentioned in the definition are primitive recursive.

Of course, every URM can be run for s steps of computation (i.e. the description of f_{m+1} from f_m in Definition 12.4.2 works for any configuration, so can be iterated for s steps). Therefore, for all $e, z, s \in \mathbb{N}$ with both $e \in URMCode$ and $z \in Seq$, there exists $p \in \mathbb{N}$ with $R(e, z, s, p)$. It follows that we can define a *total* function $f \colon \mathbb{N}^3 \to \mathbb{N}$ by

$$f(e, z, s) = \begin{cases} (\mu p\ R(e, z, s, p))[Len(p) - 1] & \text{if } e \in URMCode \text{ and } z \in Seq \\ 0 & \text{otherwise.} \end{cases}$$

That is, $f(e, z, s)$ is a configuration of the URM coded by e operating on input z after exactly s steps of computation. Notice that f is computable because R is primitive recursive. In fact, although we have used an unbounded search (which is certain to terminate) in our definition of f, it is possible to show that f is primitive recursive by providing a direct primitive recursive upper bound on the size of the smallest such p. However, since we will not need that stronger result, we leave the details as an exercise.

We now use f to define an important set H, which codes the inputs where the resulting computation halts within the given number of steps.

Definition 12.5.10. *Let*

$$H = \{(e, z, s) \in \mathbb{N}^3 : e \notin URMCode, z \notin Seq, \text{ or } f(e, z, s)[0] > Len(e)\}.$$

Also, for each $n \in \mathbb{N}^+$, *let* $H_n = \{(e, x_1, x_2, \ldots, x_n, s) \in \mathbb{N}^{n+1} : (e, \langle x_1, x_2, \ldots, x_n \rangle, s) \in H\}$.

The condition $f(e, z, s)[0] > Len(e)$ checks whether the next instruction to execute is larger than the total number of instructions of the URM coded by e. In other words, given $e \in URMCode$ and $z \in Seq$, we have $(e, z, s) \in H$ if and only if the URM coded by e operating on input z has completed its computation in at most s steps. In the cases where $e \notin URMCode$ or $z \notin Seq$, we choose to put $(e, z, s) \in H$ because there is no computation to perform, so all computation has trivially terminated. Since f is computable, we immediately obtain the following.

Proposition 12.5.11. *H is computable, and H_n is computable for each $n \in \mathbb{N}^+$.*

In fact, since f is primitive recursive (as mentioned above), the set H primitive recursive. Now that we know that can computably tell if a computation halts within a given number of steps, we move on to determine the outcome of these computations. Define a total function $g \colon \mathbb{N}^3 \to \mathbb{N}$ by

$$g(e, z, s) = \begin{cases} f(e, z, s)[1] & \text{if } e \in URMCode \text{ and } z \in Seq \\ 0 & \text{otherwise.} \end{cases}$$

In other words, when $e \in URMCode$ and $z \in Seq$, we have that $g(e, z, s)$ equals the contents of the first register after the URM coded by e runs for exactly s steps on input z. Thus, in the case where $e \in URMCode$ and $z \in Seq$, if $(e, z, s) \in H$, then we know that $g(e, z, s)$ is the output of the computation of the URM coded by e operating on input z.

Definition 12.5.12. *Let $n \in \mathbb{N}^+$ and let $e, s, x_1, x_2, \ldots, x_n \in \mathbb{N}$. We define*

$$\varphi^{(n)}_{e,s}(x_1, x_2, \ldots, x_n) = \begin{cases} g(e, \langle x_1, x_2, \ldots, x_n \rangle, s) & \text{if } H_n(e, x_1, x_2, \ldots, x_n, s) \\ \uparrow & \text{otherwise.} \end{cases}$$

Also, we use $\varphi_{e,s}$ in place of $\varphi^{(1)}_{e,s}$.

In other words, $\varphi^{(n)}_{e,s}(x_1, x_2, \ldots, x_n)$ is just like $\varphi^{(n)}_e(x_1, x_2, \ldots, x_n)$, except we cut off the computation after s steps. If the computation has not yet halted, then we output $\uparrow$. Also, notice that if $e \notin URMCode$, then $\varphi^{(n)}_{e,s}(x_1, x_2, \ldots, x_n) = 0$ for every s, which matches or definition of $\varphi^{(n)}_e(x_1, x_2, \ldots, x_n) = 0$.

The key take away is that for each $n \in \mathbb{N}^+$ the partial function

$$(e, x_1, x_2, \ldots, x_n, s) \mapsto \varphi^{(n)}_{e,s}(x_1, x_2, \ldots, x_n)$$

is in fact a partial computable function, as both g and H_n are partial computable. In other words, while the existence of a universal machine allows us to uniformly simulate any machine on any input, we can now go further and uniformly simulate any machine on any input for any given number of steps.

The following simple fact is immediate.

Proposition 12.5.13. *Let $n \in \mathbb{N}^+$, let $e, s \in \mathbb{N}$, and let $\vec{x} \in \mathbb{N}^n$.*

1. *If $\varphi^{(n)}_{e,s}(\vec{x}) \downarrow$, then for all $t > s$, we have $\varphi^{(n)}_{e,t}(\vec{x}) \downarrow$ and $\varphi^{(n)}_{e,t}(\vec{x}) = \varphi^{(n)}_{e,s}(\vec{x})$.*
2. *If $\varphi^{(n)}_{e,s}(\vec{x}) \downarrow$, then $\varphi^{(n)}_e(\vec{x}) \downarrow$ and $\varphi^{(n)}_e(\vec{x}) = \varphi^{(n)}_{e,s}(\vec{x})$.*
3. *If $\varphi^{(n)}_e(\vec{x}) \downarrow$, then there exists $s \in \mathbb{N}$ with $\varphi^{(n)}_{e,s}(\vec{x}) \downarrow$.*

Although the notation is completely standard and evocative, it is a bit misleading to write $\varphi^{(n)}_{e,s}(\vec{x}) \uparrow$, because we can computably tell whether this is the case (in contrast to $\varphi^{(n)}_e(\vec{x}) \uparrow$). Perhaps it would be better to define $\varphi^{(n)}_{e,s}(x_1, x_2, \ldots, x_n)$ to be the pair

$$\langle H_n(e, x_1, x_2, \ldots, x_n, s), g(e, \langle x_1, x_2, \ldots, x_n \rangle, s) \rangle$$

so that the first component indicates that the computation has completed, and the second holds the contents of register 1. However, this definition would conflict with the above proposition, so we do not adopt it.

With all of that background in place, we return to our original aim of showing that K is "almost" computable. We begin with an important definition.

Definition 12.5.14. *Let $A \subseteq \mathbb{N}$. We say that A is computably enumerable, or c.e., if either $A = \emptyset$ or $A = range(f)$ for some total computable function $f \colon \mathbb{N} \to \mathbb{N}$.*

For intuition, think of a total computable $f \colon \mathbb{N} \to \mathbb{N}$ as a sequence of outputs $f(0), f(1), f(2), \ldots$ of a machine. Given $b \in \mathbb{N}$, to check if $b \in A$, we want to know if b is equal to any element of this sequence. That is, to check whether $b \in A$, we compute $f(0)$ to see if it equals b, then compute $f(1)$ to see if equals b, etc. Now if $b \in A$, then this procedure eventually stops with an affirmation that b really is an element of A. However, if $b \notin A$, then this procedure never stops. In this sense, a c.e. set is "half" computable. Before establishing several equivalent ways to think about c.e. sets, we first show that the definition includes all computable sets.

Proposition 12.5.15. *Every computable subset of $\mathbb{N}$ is c.e.*

Proof. Let $A \subseteq \mathbb{N}$ be an arbitrary computable set. We have three cases:

- *Case 1:* Assume that $A = \emptyset$. We then have that A is c.e. by definition.
- *Case 2:* Suppose that A is a finite nonempty set, say $A = \{a_1, a_2, \ldots, a_n\}$. Define $f : \mathbb{N} \to \mathbb{N}$ by letting $f(k) = a_k$ for $1 \leq k \leq n$, and letting $f(k) = a_n$ otherwise. We then have that f is computable (by Proposition 12.3.14) and that $A = \text{range}(f)$.
- *Case 3:* Suppose that A is a infinite. Let a_0 be the smallest element of A. Since A is infinite, we know that for all $m \in \mathbb{N}$, there exists $b \in A$ with $b > m$. Therefore, the function $m \mapsto \mu b(b > m \wedge b \in A)$ is a total computable function. Define $f : \mathbb{N} \to \mathbb{N}$ using primitive recursion by letting

$$f(0) = a_0$$
$$f(n+1) = \mu b(b > f(n) \wedge b \in A).$$

  We then have that f is computable and that $A = \text{range}(f)$.

□

In fact, the proof of Case 3 shows something stronger: If A is an infinite computable subset of $\mathbb{N}$, then there exists a strictly increasing total computable $f : \mathbb{N} \to \mathbb{N}$ with $A = \text{range}(f)$.

Proposition 12.5.16. *Let $A \subseteq \mathbb{N}$. The following are equivalent:*

1. *A is c.e.*
2. *There exists $e \in \mathbb{N}$ such that $A = \text{range}(\varphi_e) \backslash \{\uparrow\}$.*
3. *There exists $e \in \mathbb{N}$ such that $A = \text{domain}(\varphi_e)$, i.e. such that $A = \{x \in \mathbb{N} : \varphi_e(x) \downarrow\}$.*
4. *There exists a computable $R \subseteq \mathbb{N}^2$ such that for all $x \in \mathbb{N}$, we have $x \in A \Leftrightarrow \exists y R(x, y)$.*

Before diving into the formal argument, we walk through the implication (2) $\to$ (3) at an intuitive level. Suppose then that $A = \text{range}(\varphi_e)$. We want to show that A is the domain of a partial computable function. Here is the idea. On input y, we want to halt if there exists an x for which $\varphi_e(x) = y$, and run forever otherwise. We do not know which input might produce y, so we want to compute $\varphi_e(0)$ to check it equals y, compute $\varphi_e(1)$ to check if it equals y, etc. However, unlike the case of a total function f, these computations might run forever. To remember this issue, we cut off the various computations at finite stages. But since we do not know an a priori bound on the number of stages, we have to keep pushing the bound further and further out. That is, we first compute $\varphi_{e,1}(0)$, then compute $\varphi_{e,1}(1)$, then go back to compute $\varphi_{e,2}(0)$ before returning to $\varphi_{e,2}(1)$ and then moving on to input 2. Since $\mathbb{N}^2$ is countable, we can move back and forth between all inputs and all lengths of computation. Formally, we will handle this by using codes of sequences and computing $\varphi_{e,z[0]}(z[1])$ as we increase z. This method of simulated infinitely many computations in parallel by altering the input and length of computation is known as *dovetailing*. We will follow a similar strategy for the implication (4) $\to$ (1).

Proof of Proposition 12.5.16.

- (1) $\to$ (2): Trivial

- (2) $\to$ (3): Fix $e \in \mathbb{N}$ with $A = \text{range}(\varphi_e)\backslash\{\uparrow\}$. Define $\psi\colon \mathbb{N} \to \mathbb{N}^{\uparrow}$ by letting

$$\psi(y) = \mu z(\varphi_{e,z[0]}(z[1]) \downarrow = y).$$

  We then have that ψ is a partial computable function by the above results, where we are using the fact that we can computably tell whether $\varphi_{e,z[0]}(z[1]) \downarrow$ (as $\varphi_{e,z[0]}(z[1]) \downarrow$ if and only if $H_1(e, z[1], z[0])$). Finally, notice that $y \in A \Leftrightarrow \exists x \exists s(\varphi_{e,s}(x) \downarrow = y)$ by Proposition 12.5.13, so $A = \text{domain}(\psi)$.

- (3) $\to$ (4): Fix $e \in \mathbb{N}$ with $A = \text{domain}(\varphi_e)$. Define

$$\begin{aligned} R &= \{(x, s) \in \mathbb{N}^2 : \varphi_{e,s}(x) \downarrow\} \\ &= \{(x, s) \in \mathbb{N}^2 : H_1(e, x, s)\}, \end{aligned}$$

  and notice that R is computable. For all $x \in \mathbb{N}$, we then have that $x \in A \Leftrightarrow \exists s R(x, s)$ by Proposition 12.5.13.

- (4) $\to$ (1): Suppose that $R \subseteq \mathbb{N}^2$ is computable has the property that for all x, we have $x \in A \Leftrightarrow \exists y R(x, y)$. If $A = \emptyset$, then A is c.e. by definition. Suppose then that $A \neq \emptyset$ and fix $a \in A$. Define $f\colon \mathbb{N} \to \mathbb{N}$ by letting

$$f(z) = \begin{cases} z[0] & \text{if } R(z[0], z[1]) \\ a & \text{otherwise.} \end{cases}$$

  We then have that f is a total computable function with $A = \text{range}(f)$, so A is c.e.

□

We can now argue that K is an example of a noncomputable c.e. set.

Theorem 12.5.17. *The set $K = \{e \in \mathbb{N} : \varphi_e(e) \downarrow\}$ is a c.e. set that is not computable.*

Proof. We already know that K is not computable from Theorem 12.5.7. Thus, we need only argue that K is c.e. Notice that $R = \{(e, s) \in \mathbb{N}^2 : H_1(e, e, s)\}$ is a computable relation and that $e \in K \Leftrightarrow \exists s R(e, s)$, so K is c.e. by Proposition 12.5.16. □

Also, we can prove another essential result that helps justify thinking of c.e. sets as "half" computable sets. We first recall that every computable subset of $\mathbb{N}$ is c.e. by Proposition 12.5.15. Now if $A \subseteq \mathbb{N}$ is computable, then the complement $\overline{A} = \mathbb{N}\backslash A$ is also computable, so both A and $\overline{A}$ are c.e. We now prove the converse.

Proposition 12.5.18. *Let $A \subseteq \mathbb{N}$ and suppose that both A and $\overline{A} = \mathbb{N}\backslash A$ are c.e. We then have that A is computable.*

Proof. If either $A = \emptyset$ or $A = \mathbb{N}$, this is trivial, so we may suppose that $A \neq \emptyset$ and $\overline{A} \neq \emptyset$. Fix total computable functions $f, g\colon \mathbb{N} \to \mathbb{N}$ such that $A = \text{range}(f)$ and $\overline{A} = \text{range}(g)$. Notice that for all y, there is an x such that either $f(x) = y$ or $g(x) = y$. Therefore, the function $h\colon \mathbb{N} \to \mathbb{N}$ defined by

$$h(y) = \mu x(f(x) = y \vee g(x) = y)$$

is a total computable function. Notice that $y \in A \Leftrightarrow f(h(y)) = y$. Since $\{y \in \mathbb{N} : f(h(y)) = y\}$ is computable, it follows that A is computable. □

Since K is a c.e. set that is not computable, it follows that $\overline{K} = \{e \in \mathbb{N} : \varphi_e(e) \uparrow\}$ is an example of a set that is not c.e.

Finally, we end this chapter by resolving a small puzzle that we referenced at the beginning of this section. In Proposition 12.4.23, we showed that we can generate the class of all total computable functions without ever referring to partial functions. Why does this not contradict the Church-Turing Thesis? What prevents us from using the totality of every function to computably diagonalize out? Recall at the beginning of Section 12.3 that we can perform the diagonalization argument on any class of unary functions $\mathcal{C}$ with the following properties:

1. Every element of $\mathcal{C}$ can be coded by a number (or by some finite object which can then be coded as a number) in such a way that we can decode that number and computably simulate the underlying function.
2. We can computably determine whether a given number is a valid code.
3. The functions in $\mathcal{C}$ provide an output for each input.

Now the third condition certainly holds as we are only working with total functions. Furthermore, the first condition holds here as well, since we can assign a code number to every expression that is formed by starting with the initial functions and applying $Compose$, $PrimRec$, and a restricted version of $Minimize$ that can only be used on regular functions. More formally, we can simply follow the coding that we developed in Section 12.2, but now add another clause that uses $\langle 5, a\rangle$ as a code whenever a is a valid code of a regular function with arity at least two.

Thus, if we believe the Church-Turing Thesis, then the second condition must be the issue. In fact, the set of valid codes under this scheme is *not* a computable set. After all, given a number a that codes a function of arity two (for simplicity), how can we determine whether the underlying function is regular? If the underlying function is $g\colon \mathbb{N}^2 \to \mathbb{N}$, then we need to computably determine whether for every $x \in \mathbb{N}$, there exists $y \in \mathbb{N}$ with $g(x, y) = 0$. It seems that we need to check infinitely many values of x, and perform an unbounded search for each, which looks to be a potential problem. In fact, the potentially unbounded search feels like a c.e. question, and thus seems related to the halting problem.

We do not need to be satisfied with a vague argument or with an appeal to the Church-Turing thesis. Using our work in this section, we can prove that the set of valid codes is not a computable set by showing that if we could compute it, then we could compute K. For each e, consider the total function $g_e\colon \mathbb{N}^2 \to \mathbb{N}$ defined by

$$g_e(x, s) = \begin{cases} 0 & \text{if } H_1(e, e, s) \\ 1 & \text{otherwise} \end{cases}$$

(notice that g_e ignores its first input). Although we did not prove it, we did mention above that H_1 is primitive recursive, so each g_e is primitive recursive. Moreover, since H_1 is primitive recursive, it is possible to show that there is a computable function $f\colon \mathbb{N} \to \mathbb{N}$ such that $f(e) \in PrimRecCode_2$ for each $e \in \mathbb{N}$, and such that $\Phi(f(e)) = g_e$ for all $e \in \mathbb{N}$. In other words, there is a computable function f that takes as input an $e \in \mathbb{N}$, and outputs a code for g_e. Now notice that

$$\begin{aligned} e \in K &\Leftrightarrow \exists s H_1(e, e, s) \\ &\Leftrightarrow \forall x \exists s (g_e(x, s) = 0) \\ &\Leftrightarrow g_e \text{ is regular} \\ &\Leftrightarrow \langle 5, f(e)\rangle \text{ is a valid code.} \end{aligned}$$

Thus, if we could computably tell whether a given number is a valid code, then we could compute K.

Chapter 13

Logic and Computation

Our development of a formal definition of computability in the last chapter may have seemed out of place. We used our generation template and some simple references to propositional connectives and (bounded) quantifiers, but otherwise there was seemingly little connection to logic. In this chapter, we establish that computability and logic are fundamentally intertwined. For example, assuming that we are working with a "reasonable" first-order language $\mathcal{L}$ (i.e. one where we can appropriately code symbols with natural numbers), we will show that the set of valid formulas (again suitably coded) forms a computable set. From there, we will argue that the fundamental functions related to formulas are computable, and then develop some surprising consequences that arise from our hard work on syntactic implications and the Completeness Theorem.

The connections just described are about showing that certain operation in logic are computable. However, we will also see that computability arises naturally naturally within logic itself. For example, in Chapter 14, we will give some equivalent characterizations of the computable sets defined through first-order logic, and we will show that every computable set is definable in the structure $(\mathbb{N}, 0, 1, +, \cdot, <)$. These ideas are the essential components of Gödel's famous Incompleteness Theorems.

13.1 Coding Formulas

Suppose that we are working with a "reasonable" first-order language. For the moment, consider the special case where $\mathcal{L}$ contains only finitely many constant, relation, and function symbols. In addition to these symbols in $\mathcal{L}$, we do use other symbols when creating formulas: propositional connectives, quantifiers, the equality symbol, and the variables. Since we have countably many variables, and finitely many other symbols, we can assign each symbol a code number from $\mathbb{N}$. Using our computable coding of sequences (via powers of primes), we can then assign a number to each sequence of symbols. We can then ask if the set of numbers that code valid formulas is computable.

Although most natural languages do contain only finitely many constant, relation, and function symbols, we can widen our scope to include some languages with countably many constant, relation, and function symbols. Intuitively, a language is computable if we can code the symbols as natural numbers in such a way that the sets of constant symbols, relation symbols, and function symbols are computable, and such that we can compute the arities of these functions. Before diving into the formal definition, we first fix a numbering of the logical symbols and variables that is independent of the language. We use only the even numbers in order to save the odds for the symbols of a language $\mathcal{L}$.

1. $Code(\neg) = 0$.
2. $Code(\wedge) = 2$.

3. $Code(\vee) = 4$.
4. $Code(\rightarrow) = 6$.
5. $Code(\forall) = 8$.
6. $Code(\exists) = 10$.
7. $Code(=) = 12$.
8. Fix an enumeration $\mathsf{x}_0, \mathsf{x}_1, \mathsf{x}_2, \ldots$ of Var, and let $Code(\mathsf{x}_i) = 2i + 14$ for all $i \in \mathbb{N}$.

Definition 13.1.1. *Let $\mathcal{L}$ be a (first-order) language and let $Odd = \{2n + 1 : n \in \mathbb{N}\}$. We say that $\mathcal{L}$ is a* computable language *if there exist functions $h_{\mathcal{C}} : \mathcal{C} \to Odd$, $h_{\mathcal{R}} : \mathcal{R} \to Odd$ and $h_{\mathcal{F}} : \mathcal{F} \to Odd$, together with functions $Arity_{\mathcal{R}} : \mathbb{N} \to \mathbb{N}$ and $Arity_{\mathcal{F}} : \mathbb{N} \to \mathbb{N}$, such that*

1. *The functions $h_{\mathcal{C}}$, $h_{\mathcal{R}}$, and $h_{\mathcal{F}}$ are injective.*
2. *The sets $range(h_{\mathcal{C}})$, $range(h_{\mathcal{R}})$, and $range(h_{\mathcal{F}})$ are pairwise disjoint.*
3. *The sets $range(h_{\mathcal{C}})$, $range(h_{\mathcal{R}})$, and $range(h_{\mathcal{F}})$ are computable.*
4. *$Arity_{\mathcal{R}}$ and $Arity_{\mathcal{F}}$ are computable.*
5. *$Arity_{\mathcal{R}}(n)$ is the arity of $h_{\mathcal{R}}^{-1}(n)$ for all $n \in range(h_{\mathcal{R}})$.*
6. *$Arity_{\mathcal{F}}(n)$ is the arity of $h_{\mathcal{F}}^{-1}(n)$ for all $n \in range(h_{\mathcal{F}})$.*

Suppose for the rest of this section that we are working in a computable language $\mathcal{L}$, and we have fixed such functions. In this case, we let $Code(\mathsf{c}) = h_{\mathcal{C}}(\mathsf{c})$ for all $\mathsf{c} \in \mathcal{C}$.

Definition 13.1.2. *We assign code numbers to elements of $Sym_{\mathcal{L}}^*$ by letting*

$$\sharp a_1 a_2 \cdots a_n = \langle Code(a_1), Code(a_2), \ldots, Code(a_n) \rangle.$$

We call $\sharp a_1 a_2 \cdots a_n$ the Gödel number *of the sequence $a_1 a_2 \cdots a_n$.*

Given $a \in Sym_{\mathcal{L}}$, be careful to distinguish between $Code(a)$ and $\sharp a = \langle Code(a) \rangle = 2^{Code(a)+1}$.

Definition 13.1.3. *Given $X \subseteq Sym_{\mathcal{L}}^*$, we let $\sharp X = \{\sharp \sigma : \sigma \in X\}$.*

Proposition 13.1.4. *The sets $\sharp Var$ and $\sharp Con$ are both computable.*

Proof. Notice that $\sharp Var = \{2^{2n+15} : n \in \mathbb{N}\}$ and $\sharp Con = \{2^{n+1} : n \in \text{range}(h_{\mathcal{C}})\}$, which are both computable (the latter because $\text{range}(h_{\mathcal{C}})$ is computable). □

For example, $SeqConcat(\langle\langle 2, 7\rangle, \langle 14\rangle, \langle 1, 0, 42\rangle\rangle) = \langle 2, 7, 14, 1, 0, 42\rangle$.

Proposition 13.1.5.

1. *Define a function $Concat : \mathbb{N}^2 \to \mathbb{N}$ as follows. If $w, z \in Seq$, then $Concat(w, z)$ is the number that codes the sequence obtained by concatenating the sequences coded by w and z. If either $w \notin Seq$ or $z \notin Seq$, then $Concat(w, z) = 0$. The function Concat is primitive recursive, and hence computable.*
2. *Define a function $SeqConcat : \mathbb{N} \to \mathbb{N}$ as follows. If $z \in Seq$ and $z[i] \in Seq$ for all $i < Len(z)$, then $SeqConcat(z)$ is the number that codes the sequence which is the result of concatenating the sequences coded z. If $z \notin Seq$, or if $z \in Seq$ and there exists $i < Len(z)$ with $z[i] \notin Seq$, then $SeqConcat(z) = 0$. The function SeqConcat is primitive recursive, and hence computable.*

Proof. Exercise. □

Recall that terms are defined recursively by starting with the constant symbols and variables, and then generating by putting a function symbol of arity k in front of k terms concatenated together. Given $n \in \mathbb{N}$, how can we determine whether $n \in \sharp Term_{\mathcal{L}}$? For example, suppose that $n = \langle a_0, a_1, a_2, \ldots, a_m \rangle$. If $m = 0$, then we need only check whether $n \in \sharp Con$ or $n \in \sharp Var$. Suppose then that $m \geq 1$. We first determine whether a_0 is the code of a function symbol, i.e. whether $a_0 \in \text{range}(h_{\mathcal{F}})$. If so, we can compute $Arity_{\mathcal{F}}(a_0)$. Next we want to determine whether we can break up $\langle a_1, a_2, \ldots, a_m \rangle$ into $Arity_{\mathcal{F}}(a_0)$ many pieces, each of which is an element of $Term_{\mathcal{L}}$. For example, if $m = 6$ and $Arity_{\mathcal{F}}(a_0) = 3$, then we should check whether $\langle \langle a_1 \rangle, \langle a_2, a_3, a_4 \rangle, \langle a_5, a_6 \rangle \rangle$ is a valid way to break up the sequence, i.e. whether $\langle a_1 \rangle$, $\langle a_2, a_3, a_4 \rangle$, and $\langle a_5, a_6 \rangle$ are all elements of $\sharp Term_{\mathcal{L}}$, which we can check recursively. Of course, there are several other ways to break up the sequence into 3 nonempty pieces, so we need to check whether any such partition works. Instead of trying to think about all of the ways that we can break up a sequence into a given number of pieces, it suffices to computably bound the size of any number that codes such a partition, and then use a computably bounded existential quantifier to search through all potential possibilities.

With this idea in mind, given $n = \langle a_0, a_1, \ldots, a_m \rangle$ and k, the fundamental question is how to obtain a simple computable upper bound (in terms of n and k) for all possible codes of sequences that are obtained from breaking up a sequence $\langle a_1, a_2, \ldots, a_m \rangle$ into k total pieces. For example, consider the above situation where $m = 6$ and $k = 3$, and we want a simple upper bound for

$$\langle \langle a_1 \rangle, \langle a_2, a_3, a_4 \rangle, \langle a_5, a_6 \rangle \rangle.$$

Notice that we have a sequence of length k, and each element of the sequence is strictly less than $n = \langle a_0, a_1, \ldots, a_m \rangle$. Thus, the whole sequence is strictly less than $\langle n, n, \ldots, n \rangle$, where there are m many term in the sequence.

Proposition 13.1.6. *Each of the following sets is computable:*

1. $\sharp Term_{\mathcal{L}}$.
2. $\sharp AtomicForm_{\mathcal{L}}$.
3. $\sharp Form_{\mathcal{L}}$.

Proof. Define a function $h \colon \mathbb{N}^2 \to \mathbb{N}$ by letting

$$h(n, 0) = \langle\rangle = 1$$
$$h(n, m+1) = Append(h(n, m), n)$$

for all $n, m \in \mathbb{N}$. Since *Append* is primitive recursive by Proposition 12.2.10, it follows that h is primitive recursive, and hence computable. We now turn to the

1. Define a function $f \colon \mathbb{N} \to \mathbb{N}$ using (order) recursion as follows:

$$f(n) = \begin{cases} 1 & \text{if } n \in \sharp Con \\ 1 & \text{if } n \in \sharp Var \\ 1 & \text{if } n \in Seq, Len(n) > 0, n[0] \in \text{range}(h_{\mathcal{F}}), \text{ and } (\exists z < h(n, Arity_{\mathcal{F}}(n[0]))) \\ & \qquad [z \in Seq \wedge Len(z) = Arity_{\mathcal{F}}(n[0]) \wedge (\forall i < Len(z))[z[i] < n \wedge f(z[i]) = 1] \\ & \qquad\qquad \wedge\ n = Concat(\langle n[0] \rangle, SeqConcat(z))] \\ 0 & \text{otherwise.} \end{cases}$$

   Notice that f is computable by Theorem 12.3.13 and that $f = \chi_{\sharp Term_{\mathcal{L}}}$. Therefore, $\sharp Term_{\mathcal{L}}$ is computable.

2. Define a function $g\colon \mathbb{N} \to \mathbb{N}$ using (order) recursion as follows:

$$g(n) = \begin{cases} 1 & \text{if } n \in Seq, n[0] = 12, \text{ and } (\exists a < n)(\exists b < n)[a \in \sharp Term_{\mathcal{L}} \wedge b \in \sharp Term_{\mathcal{L}} \\ & \quad \wedge\, n = SeqConcat(\langle 12 \rangle, a, b)] \\ 1 & \text{if } n \in Seq, Len(n) > 0, n[0] \in \text{range}(h_{\mathcal{R}}), \text{ and } (\exists z < h(n, Arity_{\mathcal{F}}(n[0]))) \\ & \quad [z \in Seq \wedge Len(z) = Arity_{\mathcal{R}}(n[0]) \wedge (\forall i < Len(z))[z[i] < n \wedge z[i] \in \sharp Term_{\mathcal{L}}] \\ & \quad\quad \wedge\, n = Concat(\langle n[0] \rangle, SeqConcat(z))] \\ 0 & \text{otherwise.} \end{cases}$$

Notice that g is computable by Theorem 12.3.13 and that $g = \chi_{\sharp AtomicForm_{\mathcal{L}}}$. Therefore, $\sharp AtomicForm_{\mathcal{L}}$ is computable.

3. Stuff.

□

Notice that since we can code finite sequences of numbers with numbers, we can also code finite sets of numbers with numbers. One such natural coding is as follows. Suppose that $F \subseteq \mathbb{N}$ is finite, list the elements of F in ascending order as $a_1, a_2, \ldots, a_n$, and let the code of F be $\langle a_1, a_2, \ldots, a_n \rangle$.

Proposition 13.1.7. *Suppose that we code each finite subset of $\mathbb{N}$ be the sequence that lists its elements in increasing order.*

1. *The set $SetCode \subseteq \mathbb{N}$ of all codes of finite subsets of $\mathbb{N}$ is computable.*
2. *Define $Union\colon \mathbb{N}^2 \to \mathbb{N}$ as follows. If $w, z \in SetCode$, then $Union(w, z)$ is the code of the finite set that is obtained by taking the union of the finite sets coded by w and z. Otherwise, $SetCode(w, z) = 0$. The function $Union$ is computable.*
3. *Define $Intersection\colon \mathbb{N}^2 \to \mathbb{N}$ as follows. If $w, z \in SetCode$, then $Intersection(w, z)$ is the code of the finite set that is obtained by taking the intersection of the finite sets coded by w and z. Otherwise, $SetCode(w, z) = 0$. The function $Intersection$ is computable.*
4. *Define $RelComplement\colon \mathbb{N}^2 \to \mathbb{N}$ as follows. If $w, z \in SetCode$, then $RelComplement(w, z)$ is the code of the finite set that is obtained by taking the relative complement of z inside w, i.e. the set $w \backslash z$ of elements of w that are not in z. Otherwise, $SetCode(w, z) = 0$. The function $RelComplement$ is computable.*

Definition 13.1.8. *Define a function $FreeVarCode\colon \mathbb{N} \to \mathbb{N}$ as follows. If $n \in \sharp Form_{\mathcal{L}}$, and $n = \sharp\varphi$, then $FreeVarCode(n)$ is the code of the finite set $FreeVar(\varphi)$. Otherwise, $FreeVarCode(n) = 0$.*

Proposition 13.1.9. *The following functions are computable:*

1. *$FreeVarCode\colon \mathbb{N} \to \mathbb{N}$.*
2. *$SubstCode\colon \mathbb{N}^2 \to \mathbb{N}$*
3. *$ValidSubstCode\colon \mathbb{N}^2 \to \mathbb{N}$.*

Proof. These are all defined recursively, the first one using $Union$ and $RelComplement$. □

Corollary 13.1.10. *The set $\sharp Sent_{\mathcal{L}}$ is computable.*

Since we can formulas using numbers, we can code finite sequences of formulas, and hence can code pairs (Γ, φ) as numbers. From here, we can sequences of such pairs as numbers.

We can code sequences of formulas as number, and so can code pairs (S, φ), which are just nonempty sequences of formulas, as numbers.

Proposition 13.1.11. *Let $Deduct \subseteq \mathbb{N}$ be the set of codes of such sequences which are deductions. The set $Deduct$ is computable.*

Corollary 13.1.12. *If $\Phi \subseteq Form_{\mathcal{L}}$ and $\sharp\Phi$ is computable, then the subset $Deduct_\Phi$ consisting of elements of $Deduct$ whose last line is of the form (S, φ) where $S \in \Phi^*$, is computable.*

Proposition 13.1.13. *Let $\Phi \subseteq Form_{\mathcal{L}}$ be such that $\sharp\Phi$ is computable. The set $\sharp\{\phi \in Form_{\mathcal{L}} : \Phi \vDash \phi\}$ is c.e.*

Proof. Notice that $n \in \sharp\{\phi \in Form_{\mathcal{L}} : \Phi \vDash \phi\}$ if and only if $n \in \sharp\{\phi \in Form_{\mathcal{L}} : \Phi \vdash \phi\}$ by the Soundness and Completeness Theorems, which is if and only if $\exists y(Ded_\Phi(y) \wedge Last(y) = n)$. □

Corollary 13.1.14. *If $\Sigma \subseteq Sent_{\mathcal{L}}$ and $\sharp\Sigma$ is computable, then $\sharp Cn(\Sigma)$ is c.e.*

13.2 Axiomatizable and Decidable Theories

Definition 13.2.1. *Let T be a theory in a computable language.*

1. *We say that T is* finitely axiomatizable *if there exists a finite $\Sigma \subseteq Sent_{\mathcal{L}}$ such that $T = Cn(\Sigma)$.*
2. *We say that T is* axiomatizable *if there exists $\Sigma \subseteq Sent_{\mathcal{L}}$ such that $\sharp\Sigma$ is computable and $T = Cn(\Sigma)$.*

For example, the theory Grp of groups is finitely axiomatizable, as is the theory of fields, and also the theory DLO of dense linear orderings without endpoints. Of course, in each of these cases, we *defined* the corresponding theory as the set of consequences of a finite set of sentences. For a more interesting example, notice that ACF is axiomatizable, as is ACF_p for each p. To see this, we need to consider the infinite set of sentences described in Definition 5.6.1, form the corresponding infinite set of Gödel numbers, and verify that this set is computable. If one accepts the Church-Turing thesis, then this fact follows from the intuitively clear observation that there is a mechanical procedure to perform this verification. It is possible to show that ACF (as well as each ACF_p) is not finitely axiomatizable by using Proposition 6.3.7. To do this, one needs to show that for each n, there exists a field F such that every nonconstant polynomial in $F[x]$ of degree at most n has a root in F, but F is not algebraically closed.

It is more interesting to think about theories built in the other natural way, by taking the theory of a structure. $Th(\mathbb{Q}, <)$ and $Th(\mathbb{R}, <)$ are both finitely axiomatizable, because they both equal DLO. $Th(\mathbb{C}, 0, 1, +, \cdot)$ is axiomatizable, because it equals ACF_0. What about $Th(\mathbb{R}, 0, 1, +, \cdot)$ or $Th(\mathbb{N}, 0, 1, +, \cdot)$?

Axiomatizable: Can I write down a nice set of sentences that captures everything in the theory?

Definition 13.2.2. *We say that a theory T is* decidable *if $\sharp T$ is computable.*

Decidable: Can I devise a mechanical procedure that will determine whether a given sentence is in the theory?

Proposition 13.2.3. *Let T be a theory in a computable language.*

1. *If T is decidable, then T is axiomatizable.*
2. *If T is axiomatizable, then $\sharp T$ is c.e.*
3. *If T is both axiomatizable and complete, then T is decidable.*

Proof. 1. Just take T itself as the axioms.

2. Cite result in previous section.

3. Use fact that if a set and its complement are both c.e., then the set is computable.

□

Corollary 13.2.4. *DLO is decidable.*

Corollary 13.2.5. *ACF_p is decidable for each p.*

Chapter 14

Incompleteness

Let $\mathcal{L} = \{0, 1, +, \cdot, <\}$ where 0 and 1 are constant symbols, $+$ and $\cdot$ are binary function symbols, and $<$ is a binary relation symbol. Consider the $\mathcal{L}$-structure $\mathcal{N} = (\mathbb{N}, 0, 1, +, \cdot, <)$. In Section 4.4, we alluded to fact that $\mathcal{N}$ has many interesting and complicated definable sets. In this chapter, we will show that every computable set is definable in $\mathcal{N}$, and use this fact to prove that $Th(\mathcal{N})$ is not axiomatizable, which is a weak version of Gödel's First Incompleteness Theorem. We will then expand these ideas to a wider context in order to establish each of Gödel's Incompleteness Theorems.

14.1 Definability in Arithmetic

Definition 14.1.1. *Let $\mathcal{L} = \{0, 1, +, \cdot, <\}$ where 0 and 1 are constant symbols, $+$ and $\cdot$ are binary function symbols, and $<$ is a binary relation symbol. Let $\mathcal{N} = (\mathbb{N}, 0, 1, +, \cdot, <)$.*

The structure $\mathcal{N}$ has many interesting definable sets. For example, the set of evens is defined by

$$\exists y(y + y = x)$$

and the set of primes is defined by

$$\neg(x = 1) \wedge \forall y \forall z(x = y \cdot z \rightarrow (y = 1 \vee z = 1)).$$

Surprisingly, as alluded to in Section 4.4, every computable subset of $\mathbb{N}^n$ is definable in $\mathcal{N}$. To prove this, we first show that the graph of every total computable function is definable in $\mathcal{N}$ (since we know how to generate the total computable functions). To this end, recall Proposition 12.4.23, which says that the total computable functions can obtained by starting with $Init$, and generating using $Compose$, $PrimRec$, and a restricted version of $Minimize$ that can only be applied only to regular functions. It is easy to see that the graph of any function in $Init$ is definable in $\mathcal{N}$, so it suffices to show that the collection of total functions whose graph is definable in $\mathcal{N}$ is closed under $Compose$, $PrimRec$, and our restricted version of $Minimize$. Now it is reasonably straightforward to prove closure under $Compose$ and $Minimize$, but $PrimRec$ is much more difficult. To get around this obstacle, we prove the following amazing fact, which that if we add a few more basic functions to $Init$, then we can generate all total computable functions without using $PrimRec$.

Theorem 14.1.2 (Essentially Gödel)**.** *The class of total computable functions is the collection of total functions obtained by starting with $\mathcal{O}$, $\mathcal{S}$, $\mathcal{I}_i^n$, $+$, $\cdot$, and Equal, and closing off under Compose and applications of Minimize restricted to regular functions.*

For the interesting direction, we need to show that the class $\mathcal{G}$ of functions described in the theorem is closed under $PrimRec$. That is, if $g\colon \mathbb{N}^n \to \mathbb{N}$ and $h\colon \mathbb{N}^{n+2} \to \mathbb{N}$ are both in $\mathcal{G}$, then $f = PrimRec(g, h) \in \mathcal{G}$.

How can we do this using only *Compose* and *Minimize*? We start by remembering how to compute $f(\vec{x}, y)$ given procedures to compute g and h. Intuitively, we first compute $f(\vec{x}, 0) = g(\vec{x})$, next use this to compute $f(\vec{x}, 1) = h(\vec{x}, 0, f(\vec{x}, 0))$, then use this to compute $f(\vec{x}, 2) = h(\vec{x}, 1, f(\vec{x}, 1))$, etc. Now given a tuple $\vec{x}$ and a number y, we can code up the sequence of values $\langle f(\vec{x}, 0), f(\vec{x}, 1), \ldots, f(\vec{x}, y)\rangle$ that result from this computation. Notice that this sequence is coded by one number z that has the "course-of-values" of f stored in its components.

Let's think about this coding from the other direction. That is, given a tuple $\vec{x}$ and number z, we can determine whether z codes a sequence $\langle a_0, a_1, \ldots, a_n\rangle$ with $a_0 = g(\vec{x})$ and such that $a_{i+1} = h(\vec{x}, i, a_i)$ for each i. If so, then we know that $a_i = f(\vec{x}, i)$ for each i. Thus, to determine $f(\vec{x}, y)$ without directly using primitive recursion, we can simply search, using *Minimize*, for the smallest number z that codes a correct computation sequence (in the sense just described) of length at least $y + 1$, and then pull out the entry in position $y + 1$.

Although this is a promising idea, the primary obstacle is the fact that our coding and decoding of sequences used primitive recursion (because we used powers of primes, and both exponentiation and the function that enumerates the primes in Proposition 12.1.26 were both defined using primitive recursion). As a result, the most interesting part of our argument will be to show that $\mathcal{G}$ contains a function $\beta\colon \mathbb{N}^2 \to \mathbb{N}$ that can find any particular finite sequence "encoded" within some number. More formally, β will have the property that for all $a_0, a_1, \ldots, a_n \in \mathbb{N}$, there exists $c \in \mathbb{N}$ such that $\beta(c, i) = a_i$ for all $i \leq n$. Note that *DecodeSeq* has this property.

Let's examine how we can use the existence of such a function $\beta \in \mathcal{G}$ to carry out the about outline. Suppose that $g\colon \mathbb{N}^n \to \mathbb{N}$ and $h\colon \mathbb{N}^{n+2} \to \mathbb{N}$ are both in $\mathcal{G}$. Let $f = PrimRec(g, h) \in \mathcal{G}$. Define a function $t\colon \mathbb{N}^{n+1} \to \mathbb{N}$ by letting

$$t(\vec{x}, y) = \mu z[\beta(z, 0) = g(\vec{x}) \wedge (\forall i < y)(\beta(z, i + 1) = h(\vec{x}, i, \beta(z, i)))].$$

Now assuming that the relations with characteristic function in $\mathcal{G}$ are closed under a few basic operations (like $\wedge$ and bounded quantification), it will follow that $t \in \mathcal{G}$. Since $f(\vec{x}, y) = \beta(t(\vec{x}, y), y)$, we will then be able to conclude that $f \in \mathcal{G}$.

In fact, it will be difficult to build β directly, so we will instead show that $\mathcal{G}$ contains a function $\alpha\colon \mathbb{N}^3 \to \mathbb{N}$ that is able to find any particular finite sequence "encoded" within some *pair* of numbers. That is, α will have the property that for all $a_0, a_1, \ldots, a_n \in \mathbb{N}$, there exists $b, k \in \mathbb{N}$ such that $\alpha(b, k, i) = a_i$. The idea behind α is to get the a_i's as remainders upon division of the number b by $n + 1$ many numbers defined in terms of k and i. The key fact that we will use is the Chinese Remainder Theorem from number theory.

Theorem 14.1.3 (Chinese Remainder Theorem)**.** *Let $m_0, m_1, \ldots, m_n \in \mathbb{N}^+$ be pairwise relatively prime, and let $a_0, a_1, \ldots, a_n \in \mathbb{N}$. There exists $b \in \mathbb{N}$ such that $b \equiv a_i \pmod{m_i}$ for all i.*

We now carry out the details. We will first establish that $\mathcal{G}$ has necessary closure properties, then move on to proving the existence of an $\alpha \in \mathcal{G}$ with the above properties, and then use it to construct β by arguing that $\mathcal{G}$ contains a pairing function along with suitable decodings it.

Proof of Theorem 14.1.2. Let $\mathcal{G}$ be the collection of all functions obtained by starting with $\mathcal{O}$, $\mathcal{S}$, $\mathcal{I}_i^n$, $+$, $\cdot$, and *Equal*, and closing off under *Compose* and applications of *Minimize* restricted to regular functions. Since $+$, $\cdot$, and *Equal* are all total computable functions, and the class of total computable functions is closed under *Compose* and the restricted version of *Minimize*, it follows that every element of $\mathcal{G}$ is computable.

We now prove that every computable function is in $\mathcal{G}$ by arguing that $\mathcal{G}$ is closed under *PrimRec* and appealing to Proposition 12.4.23. We first argue that $\mathcal{G}$ contains a few simple functions and has several important closure properties. In the rest of the argument, given a relation $R \subseteq \mathbb{N}^n$, we write $R \in \mathcal{G}$ to mean that $\chi_R \in \mathcal{G}$.

1. The constant functions $\mathcal{C}_k^n\colon \mathbb{N}^n \to \mathbb{N}$ defined by $\mathcal{C}_k^n(\vec{x}) = k$ are all in $\mathcal{G}$: See the proof of Proposition 12.1.6, which only uses the basic functions and *Compose*.

2. The functions $isZero, isNonzero\colon \mathbb{N} \to \mathbb{N}$ are both in $\mathcal{G}$: Notice that $isZero = Compose(Equal, \mathcal{I}_1^1, \mathcal{C}_0^1)$, so $isZero \in \mathcal{G}$. Since $isNonzero = Compose(isZero, isZero)$, it follows that $isNonzero \in \mathcal{G}$.

3. Relations in $\mathcal{G}$ are closed under complement, union, and intersection: See the proof of Proposition 12.1.15, which only uses $+$, $\cdot$, $isZero$, $isNonzero$, and $Compose$.

4. If $n \in \mathbb{N}^+$ and $R \subseteq \mathbb{N}^{n+1}$ is a relation in $\mathcal{G}$ with the property that for all $\vec{x} \in \mathbb{N}^n$, there exists $y \in \mathbb{N}$ with $R(\vec{x}, y)$, then the function $h(\vec{x}) = \mu z R(\vec{x}, z)$ is in $\mathcal{G}$: Define $g\colon \mathbb{N}^{n+1} \to \mathbb{N}$ by letting

$$g(\vec{x}, y) = \begin{cases} 0 & \text{if } R(\vec{x}, y) \\ 1 & \text{otherwise.} \end{cases}$$

Notice that $g = Compose(isNonzero, \chi_R)$, so $g \in \mathcal{G}$. Also, g is regular by our assumption about R. Since $h = Minimize(g)$, it follows that $h \in \mathcal{G}$.

5. If $n \in \mathbb{N}^+$, $R \subseteq \mathbb{N}^{n+1}$ is a relation in $\mathcal{G}$, and $g\colon \mathbb{N}^{n+1} \to \mathbb{N}$ is in $\mathcal{G}$, then the function $f\colon \mathbb{N}^{n+1} \to \mathbb{N}$ defined by $f(\vec{x}, y) = (\mu z < g(\vec{x}, y))R(\vec{x}, z)$ is in $\mathcal{G}$: Let

$$S = \{(\vec{x}, y, z) \in \mathbb{N}^{n+2} : R(\vec{x}, z) \vee z = g(\vec{x}, y)\}.$$

Since $g, Equal \in \mathcal{G}$, and relations in $\mathcal{G}$ are closed under union, it follows that $S \in \mathcal{G}$. Moreover, notice that for all $(\vec{x}, y) \in \mathbb{N}^{n+1}$, there exists $z \in \mathbb{N}$ with $S(\vec{x}, y, z)$. Since $f(\vec{x}, y) = \mu z S(\vec{x}, y, z)$ for all $(\vec{x}, y) \in \mathbb{N}$, it follows from (4) that $f \in \mathcal{G}$.

6. If $n \in \mathbb{N}^+$, $R \subseteq \mathbb{N}^{n+1}$ is a relation in $\mathcal{G}$, and $g\colon \mathbb{N}^{n+1} \to \mathbb{N}$ is in $\mathcal{G}$, then the relations

$$S_1 = \{(\vec{x}, y) \in \mathbb{N}^{n+1} : (\exists z < g(\vec{x}, y))R(\vec{x}, z)\}$$
$$S_2 = \{(\vec{x}, y) \in \mathbb{N}^{n+1} : (\forall z < g(\vec{x}, y))R(\vec{x}, z)\}$$

are both in $\mathcal{G}$: The function $f\colon \mathbb{N}^{n+1} \to \mathbb{N}$ defined by $f(\vec{x}, y) = (\mu z < g(\vec{x}, y))R(\vec{x}, z)$ is in $\mathcal{G}$ by (5). Notice that $(\vec{x}, y) \in S_1$ if and only if $f(\vec{x}, y) \neq g(\vec{x}, y)$. Since $g, Equal \in \mathcal{G}$, and relations in $\mathcal{G}$ are closed under complement, it follows that $S_1 \in \mathcal{G}$. To see that $S_2 \in \mathcal{G}$, simply observe that

$$S_2 = \{(\vec{x}, y) \in \mathbb{N}^{n+1} : \neg(\exists z < g(\vec{x}, y))\neg R(\vec{x}, z)\},$$

and use the fact that the relations in $\mathcal{G}$ are closed under complement and bounded existential quantification (by what we just showed).

7. $\mathcal{G}$ contains a pairing function along with corresponding decoding functions: At the beginning of Section 12.2, we showed that the usual bijection from $\mathbb{N}^2$ to $\mathbb{N}$ obtained by walking down the finite diagonals of $\mathbb{N}^2$ is primitive recursive. Letting $Pair\colon \mathbb{N}^2 \to \mathbb{N}$ be this function, we argued that

$$Pair(x, y) = (\mu z < (x^2 + y^2 + 2xy + 3x + y + 1))(2z = x^2 + y^2 + 2xy + 3x + y).$$

Since $+, \cdot, Equal \in \mathcal{G}$, and $\mathcal{G}$ is closed under the bounded μ-operator by (5), it follows that $Pair \in \mathcal{G}$. Letting $Left, Right\colon \mathbb{N} \to \mathbb{N}$ be the corresponding decoding functions, we argued that

$$Left(z) = (\mu x < z + 1)[(\exists y < z + 1)(Pair(x, y) = z)]$$
$$Right(z) = (\mu y < z + 1)[(\exists x < z + 1)(Pair(x, y) = z)],$$

so $Left, Right \in \mathcal{G}$ by (5) and (6).

Define $\alpha\colon \mathbb{N}^3 \to \mathbb{N}$ by letting $\alpha(b,k,i)$ be the remainder upon division of b by $1+(i+1)k$. To see that $\alpha \in \mathcal{G}$, simply notice that

$$\alpha(b,k,i) = (\mu r < b)[(\exists q < b+1)(b = q\cdot(1+(i+1)k)+r)],$$

and use the closure properties of $\mathcal{G}$ established above. We claim that for all $a_0, a_1, \ldots, a_n \in \mathbb{N}$, there exists $b,k \in \mathbb{N}$ such that $\alpha(b,k,i) = a_i$. Let $a_0, a_1, \ldots, a_n \in \mathbb{N}$ be arbitrary. Let $s = \max\{n, a_0, a_1, \ldots, a_n\}+1$ and let $k = s!$. We first argue that the numbers $1+k, 1+2k, \ldots, 1+(n+1)k$ are pairwise relatively prime. Let i,j be arbitrary with $1 \leq i,j \leq n+1$. Suppose that p is prime, and that both $p \mid (1+ik)$ and $p \mid (1+jk)$. Since $p \mid (1+ik)$, we must have $p \nmid k$, hence it must be the case that $p > s$. We also have $p \mid (i-j)k$, so since p is prime and $p \nmid k$, it follows that $p \mid (i-j)$. As $p > s \geq n+1$, it must be the case that $i = j$. Thus, the numbers $1+k, 1+2k, \ldots, 1+(n+1)k$ are pairwise relatively prime. Since $a_i < s \leq k < 1+(i+1)k$ for all k, it follows by the Chinese Remainder Theorem that there exists $b \in \mathbb{N}$ such that $\alpha(b,k,i) = a_i$ for all i.

Now define $\beta\colon \mathbb{N}^2 \to \mathbb{N}$ by letting $\beta(c,i) = \alpha(Left(c), Right(c), i)$. Since $Left, Right, \alpha \in \mathcal{G}$, it follows that $\beta \in \mathcal{G}$. Let $a_0, a_1, \ldots, a_n \in \mathbb{N}$ be arbitrary. From above, we can fix $b,k \in \mathbb{N}$ with $\alpha(b,k,i) = a_i$ for all i. Letting $c = Pair(b,k)$, we then have that

$$\begin{aligned}\beta(c,i) &= \alpha(Left(c), Right(c), i)\\ &= \alpha(Left(Pair(b,k)), Right(Pair(b,k)), i)\\ &= \alpha(b,k,i)\\ &= a_i\end{aligned}$$

for all $i \leq n$.

We now show that $\mathcal{G}$ is closed under $PrimRec$, Let $n \in \mathbb{N}^+$, and let $g\colon \mathbb{N}^n \to \mathbb{N}$ and $h\colon \mathbb{N}^{n+2} \to \mathbb{N}$ be such that $g, h \in \mathcal{G}$. Let $f = PrimRec(g,h)$. Define $R \subseteq \mathbb{N}^{n+2}$ by

$$R = \{(\vec{x}, y, z) \in \mathbb{N}^{n+2} : \beta(z,0) = g(\vec{x}) \wedge (\forall i < y)(\beta(z, i+1) = h(\vec{x}, i, \beta(z,i)))\}$$

Since $\beta, g, h \in \mathcal{G}$, and $\mathcal{G}$ is closed under boolean operations and bounded quantification, it follows that $R \in \mathcal{G}$. Moreover, using the just established property of β (that for all $a_0, a_1, \ldots, a_n \in \mathbb{N}$, there exists $c \in \mathbb{N}$ with $\beta(c,i) = a_i$ for all i), we conclude that for all $(\vec{x}, y) \in \mathbb{N}^{n+1}$, there exists $z \in \mathbb{N}$ with $(\vec{x}, y, z) \in R$. Using closure property (4) of $\mathcal{G}$ established above, the function $t\colon \mathbb{N}^{n+1} \to \mathbb{N}$ defined by

$$t(\vec{x}, y) = \mu z[\beta(z,0) = g(\vec{x}) \wedge (\forall i < y)(\beta(z, i+1) = h(\vec{x}, i, \beta(z,i)))]$$

is an element of $\mathcal{G}$. Since $f(\vec{x}, y) = \beta(t(\vec{x}, y), y)$, it follows that $f \in \mathcal{G}$.

□

Theorem 14.1.4. *The graph of every total computable function is definable in $\mathcal{N}$.*

Proof. We use Theorem 14.1.2. We first handle the basic functions:

- The graph of $\mathcal{O}$ is defined by the formula $\mathsf{x} = \mathsf{x} \wedge \mathsf{y} = 0$.
- The graph of $\mathcal{S}$ is defined by the formula $\mathsf{y} = \mathsf{x} + 1$.
- The graph of $\mathcal{I}_i^n$ is defined by the formula $\mathsf{x_1} = \mathsf{x_1} \wedge \mathsf{x_2} = \mathsf{x_2} \wedge \cdots \wedge \mathsf{x_n} = \mathsf{x_n} \wedge \mathsf{y} = \mathsf{x_i}$.
- The graph of $+$ is defined by the formula $\mathsf{y} = \mathsf{x_1} + \mathsf{x_2}$.
- The graph of $\cdot$ is defined by the formula $\mathsf{y} = \mathsf{x_1} \cdot \mathsf{x_2}$.
- The graph of $Equal$ is defined by the formula $(\mathsf{x_1} = \mathsf{x_2} \wedge \mathsf{y} = 1) \vee (\neg(\mathsf{x_1} = \mathsf{x_2}) \wedge \mathsf{y} = 0)$.

We now argue that the collection of functions whose graph is definable in $\mathcal{N}$ is closed under *Compose* and *Minimize* restricted to regular functions:

- Suppose that $h\colon \mathbb{N}^m \to \mathbb{N}$ is definable in $\mathcal{N}$ and that $g_1, g_2, \ldots, g_m\colon \mathbb{N}^n \to \mathbb{N}$ are definable in $\mathcal{N}$. Let $f = Compose(h, g_1, g_2, \ldots, g_m)$. Fix $\psi(\mathsf{y}_1, \mathsf{y}_2, \ldots, \mathsf{y}_\mathsf{m}, \mathsf{z}) \in Form_{\mathcal{L}}$ defining $graph(h)$ in $\mathcal{N}$ and fix $\theta_\mathsf{i}(\mathsf{x}_1, \mathsf{x}_2, \ldots, \mathsf{x}_\mathsf{n}, \mathsf{y}) \in Form_{\mathcal{L}}$ defining $graph(g_i)$ in $\mathcal{N}$. The formula $\varphi(\mathsf{x}_1, \mathsf{x}_2, \ldots, \mathsf{x}_\mathsf{n}, \mathsf{z}) \in Form_{\mathcal{L}}$ given by

$$\exists \mathsf{y}_1 \exists \mathsf{y}_2 \cdots \exists \mathsf{y}_\mathsf{m}((\bigwedge_{\mathsf{i}=1}^{\mathsf{m}} \theta_\mathsf{i}(\mathsf{x}_1, \mathsf{x}_2, \ldots, \mathsf{x}_\mathsf{n}, \mathsf{y}_\mathsf{i})) \wedge \psi(\mathsf{y}_1, \mathsf{y}_2, \ldots, \mathsf{y}_\mathsf{m}, \mathsf{z})).$$

  then defines $graph(f)$ in $\mathcal{N}$.

- Suppose that $g\colon \mathbb{N}^{n+1} \to \mathbb{N}$ is a regular function that is definable in $\mathcal{N}$. Let $f = Minimize(g)$. Fix $\psi(\mathsf{x}_1, \mathsf{x}_2, \ldots, \mathsf{x}_\mathsf{n}, \mathsf{y}, \mathsf{z}) \in Form_{\mathcal{L}}$ defining $graph(g)$ in $\mathcal{N}$. The formula $\varphi(\mathsf{x}_1, \mathsf{x}_2, \ldots, \mathsf{x}_\mathsf{n}, \mathsf{z}) \in Form_{\mathcal{L}}$ given by

$$\psi(\mathsf{x}_1, \mathsf{x}_2, \ldots, \mathsf{x}_\mathsf{n}, \mathsf{y}, 0) \wedge \forall \mathsf{w}(\mathsf{w} < \mathsf{y} \to \neg\psi(\mathsf{x}_1, \mathsf{x}_2, \ldots, \mathsf{x}_\mathsf{n}, \mathsf{w}, 0)).$$

  then defines $graph(f)$ in $\mathcal{N}$.

Using induction and Theorem 14.1.2, it follows that the graph of every computable function is definable in $\mathcal{N}$. □

Corollary 14.1.5. *Every computable relation $R \subseteq \mathbb{N}^n$ is definable in $\mathcal{N}$.*

Proof. Let $R \subseteq \mathbb{N}^n$ be an arbitrary computable relation. By definition, this means that the characteristic function $\chi_R\colon \mathbb{N}^n \to \mathbb{N}$ is computable. By Theorem 14.1.4, we can fix $\varphi(\mathsf{x}_1, \mathsf{x}_2, \ldots, \mathsf{x}_\mathsf{n}, \mathsf{y}) \in Form_{\mathcal{L}}$ defining $graph(\chi_R)$ in $\mathcal{N}$. We then have that the formula $\varphi(\mathsf{x}_1, \mathsf{x}_2, \ldots, \mathsf{x}_\mathsf{n}, 1)$ defines R in $\mathcal{N}$. □

Corollary 14.1.6. *If $A \subseteq \mathbb{N}$ is c.e., then A is definable in $\mathcal{N}$.*

Proof. Since A is c.e., we can use Proposition 12.5.16 to fix a computable set $R \subseteq \mathbb{N}^2$ such that

$$m \in A \Leftrightarrow \text{There exists } n \in \mathbb{N} \text{ with } (m, n) \in R.$$

Now R is computable, so it is definable in $\mathcal{N}$ by Corollary 14.1.5. Fix $\psi(\mathsf{x}, \mathsf{y}) \in Form_{\mathcal{L}}$ defining R in $\mathcal{N}$. Letting $\varphi(\mathsf{x})$ be the formula $\exists \mathsf{y}\psi(\mathsf{x}, \mathsf{y})$, we then have that φ defines A in $\mathcal{N}$. □

14.2 Incompleteness and Undecidability

Our goal in this section is to prove the following theorem which is a weak form of the First Incompleteness Theorem. We will strengthen it later, but all of the real insight and hard work is in this version anyway.

Theorem 14.2.1 (First Incompleteness Theorem - Gödel)**.** *$Th(\mathcal{N})$ is undecidable and not axiomatizable.*

Notice that since $Th(\mathcal{N})$ is a complete theory, we know that it is undecidable if and only if it is not axiomatizable. Thus, it suffices to prove only one. We will give three proofs below.

Proof Using the Halting Problem (or any Noncomputable C.E. Set)

Proof of Incompleteness Theorem via Computability. Suppose that $\sharp\Sigma$ is computable and that $Cn(\Sigma) = Th(\mathcal{N})$. By Theorem 12.5.17, we can fix a c.e. set K that is not computable (for example, we can take $K = \{e \in \mathbb{N} : \varphi_e(e)\downarrow\}$). Using Corollary 14.1.6, fix a formula $\varphi(\mathsf{x})$ defining K in $\mathcal{N}$. Notice that the set $\{n \in \mathbb{N} : \Sigma \vDash \neg\varphi(\underline{n})\}$ is c.e. However, we have that

$$\begin{aligned} n \in \overline{K} &\Leftrightarrow \mathcal{N} \vDash \neg\varphi(\underline{n}) \\ &\Leftrightarrow \Sigma \vDash \neg\varphi(\underline{n}), \end{aligned}$$

so $\overline{K} = \{n \in \mathbb{N} : \Sigma \vDash \varphi(\underline{n})\}$ is c.e., contradicting Proposition 12.5.18 (which says that if both A and $\overline{A}$ are c.e., then A is computable). □

In the next two proofs we will make use of the function $f\colon \mathbb{N}^2 \to \mathbb{N}$ that thinks of the first argument as the code of formula with at most one free variable, and plugs in the numeral given by the second argument in for the variable. That is,

$$f(m,n) = \begin{cases} \sharp\varphi(\underline{n}) & \text{if } m = \sharp\varphi \text{ for a formula } \varphi \text{ with one free variable} \\ 0 & \text{otherwise} \end{cases}$$

Proof Using Undefinabilty of Truth

Theorem 14.2.2 (Undefinability of Truth - Tarski)**.** *The set $\sharp Th(\mathcal{N})$ is not definable in $\mathcal{N}$.*

Big idea: Suppose that $\sharp Th(\mathcal{N})$ was definable in $\mathcal{N}$. Fix a formula $\alpha(\mathsf{x})$ defining it. By Theorem 14.1.4, we can fix $\theta(\mathsf{x},\mathsf{y},\mathsf{z})$ defining the graph of f in $\mathcal{N}$. The key is to use α and θ to build a definable set $R \subseteq \mathbb{N}^2$ such that every definable subset of $\mathbb{N}$ appears as a row. This would then allow us to definably diagonalize out of the definable sets (essentially by taking the negation of the diagonal), which leads to a contradiction.

Proof. Suppose that $\{\sharp\sigma : \mathcal{N} \vDash \sigma\}$ is definable in $\mathcal{N}$, and fix $\alpha(\mathsf{x}) \in Form_{\mathcal{L}}$ defining it. We then have

$$\mathcal{N} \vDash \sigma \Leftrightarrow \mathcal{N} \vDash \alpha(\underline{\sharp\sigma})$$

for all $\sigma \in Sent_{\mathcal{L}}$. By Theorem 14.1.4, we can fix $\theta(\mathsf{x},\mathsf{y},\mathsf{z})$ defining the graph of f in $\mathcal{N}$. Let $\psi(\mathsf{x},\mathsf{y})$ be the formula $\forall\mathsf{z}(\theta(\mathsf{x},\mathsf{y},\mathsf{z}) \to \alpha(\mathsf{z}))$. Notice that for all $\varphi(\mathsf{x}) \in Form_{\mathcal{L}}$ and all $n \in \mathbb{N}$, we have

$$\begin{aligned} \mathcal{N} \vDash \psi(\underline{\sharp\varphi},\underline{n}) &\Leftrightarrow \mathcal{N} \vDash \alpha(\underline{f(\sharp\varphi,n)}) \\ &\Leftrightarrow \mathcal{N} \vDash \alpha(\underline{\sharp\varphi(\underline{n})}) \\ &\Leftrightarrow \mathcal{N} \vDash \varphi(\underline{n}). \end{aligned}$$

Now let $\varphi(\mathsf{x})$ be the formula $\neg\psi(\mathsf{x},\mathsf{x})$. We then have

$$\begin{aligned} \mathcal{N} \vDash \varphi(\underline{\sharp\varphi}) &\Leftrightarrow \mathcal{N} \vDash \neg\psi(\underline{\sharp\varphi},\underline{\sharp\varphi}) \\ &\Leftrightarrow \mathcal{N} \nvDash \psi(\underline{\sharp\varphi},\underline{\sharp\varphi}) \\ &\Leftrightarrow \mathcal{N} \nvDash \varphi(\underline{\sharp\varphi}), \end{aligned}$$

which is a contradiction. □

Proof of Incompleteness Theorem via Definability. If $Th(\mathcal{N})$ is decidable, then $\sharp Th(\mathcal{N})$ is computable, hence definable in $\mathcal{N}$, contradicting Undefinability of Truth. Therefore, $Th(\mathcal{N})$ is undecidable. □

Proof Using a Sentence Implying Its Nonprovability

Our next proof of Incompleteness uses the following fundamental lemma which allows us to make sentences that indirectly refer to themselves.

Lemma 14.2.3 (Fixed-Point Lemma - Gödel)**.** *Let $\alpha(\mathsf{x}) \in Form_{\mathcal{L}}$. There exists $\sigma \in Sent_{\mathcal{L}}$ such that*

$$\mathcal{N} \vDash \sigma \leftrightarrow \alpha(\underline{\sharp\sigma}).$$

Proof. As above, fix $\theta(\mathsf{x},\mathsf{y},\mathsf{z})$ defining the graph of f in $\mathcal{N}$ and let $\psi(\mathsf{x},\mathsf{y})$ be the formula $\forall\mathsf{z}(\theta(\mathsf{x},\mathsf{y},\mathsf{z}) \to \alpha(\mathsf{z}))$. Notice that for all $\varphi(\mathsf{x}) \in Form_{\mathcal{L}}$ and all $n \in \mathbb{N}$, we have

$$\begin{aligned}\mathcal{N} \vDash \psi(\underline{\sharp\varphi}, \underline{n}) &\Leftrightarrow \mathcal{N} \vDash \alpha(\underline{f(\sharp\varphi, n)}) \\ &\Leftrightarrow \mathcal{N} \vDash \alpha(\underline{\sharp\varphi(\underline{n})}).\end{aligned}$$

Now let $\varphi(\mathsf{x})$ be the formula $\psi(\mathsf{x},\mathsf{x})$ (so φ defines the diagonal). The point here is to look at what happens when the row $\sharp\varphi$ which is defining the diagonal actually meets the diagonal. That is, we should look at the $(\sharp\varphi, \sharp\varphi)$ entry of the table. We have

$$\begin{aligned}\mathcal{N} \vDash \varphi(\underline{\sharp\varphi}) &\Leftrightarrow \mathcal{N} \vDash \psi(\underline{\sharp\varphi}, \underline{\sharp\varphi}) \\ &\Leftrightarrow \mathcal{N} \vDash \alpha(\underline{\sharp\varphi(\underline{\sharp\varphi})})\end{aligned}$$

Thus, if we let $\sigma = \varphi(\underline{\sharp\varphi})$, we then have that $\mathcal{N} \vDash \sigma$ if and only if $\mathcal{N} \vDash \alpha(\underline{\sharp\sigma})$. That is, $\mathcal{N} \vDash \sigma \leftrightarrow \alpha(\underline{\sharp\sigma})$. □

We first show how to get Undefinability of Truth using the Fixed-Point Lemma. The idea is to take a purported definition of truth, and use it to get a sentence σ which indirectly says that it is false.

Using the Fixed-Point Lemma to Prove Undefinability of Truth. Suppose that the set $\{\sharp\sigma : \mathcal{N} \vDash \sigma\}$ is definable in $\mathcal{N}$, and fix $\alpha(\mathsf{x}) \in Form_{\mathcal{L}}$ defining it so that

$$\mathcal{N} \vDash \sigma \Leftrightarrow \mathcal{N} \vDash \alpha(\underline{\sharp\sigma})$$

for all $\sigma \in Sent_{\mathcal{L}}$. By the Fixed-Point Lemma, there exists $\sigma \in Sent_{\mathcal{L}}$ such that $\mathcal{N} \vDash \sigma \leftrightarrow \neg\alpha(\underline{\sharp\sigma})$. We then have that

$$\begin{aligned}\mathcal{N} \vDash \alpha(\underline{\sharp\sigma}) &\Leftrightarrow \mathcal{N} \vDash \sigma \\ &\Leftrightarrow \mathcal{N} \vDash \neg\alpha(\underline{\sharp\sigma})\end{aligned}$$

a contradiction. □

We now give another proof of incompleteness using a sentence which indirectly asserts that it is not provable.

Proof of Incompleteness Theorem via Self-Reference. Suppose that $\Sigma \subseteq Th(\mathcal{N})$ and that $\sharp\Sigma$ is computable. We then have that the set $\{\sharp\sigma : \Sigma \vDash \sigma\}$ is c.e., so is definable in $\mathcal{N}$ by Corollary 14.1.6. Fix $Prv_{\Sigma}(\mathsf{x}) \in Form_{\mathcal{L}}$ defining $\{\sharp\sigma : \Sigma \vDash \sigma\}$ in $\mathcal{N}$, so that

$$\mathcal{N} \vDash Prv_{\Sigma}(\underline{\sharp\sigma}) \Leftrightarrow \Sigma \vDash \sigma$$

for all $\sigma \in Sent_{\mathcal{L}}$. By the Fixed-Point Lemma, there exists $\sigma \in Sent_{\mathcal{L}}$ such that

$$\mathcal{N} \vDash \sigma \leftrightarrow \neg Prv_{\Sigma}(\underline{\sharp\sigma}).$$

Now if $\Sigma \models \sigma$, we would then have that $\mathcal{N} \models \sigma$ (because $\Sigma \subseteq Th(\mathcal{N})$), but we also have

$$\begin{aligned} \Sigma \models \sigma &\Rightarrow \mathcal{N} \models Prv_\Sigma(\underline{\sigma}) \\ &\Rightarrow \mathcal{N} \models \neg\sigma, \end{aligned}$$

which is a contradiction. Therefore, we must have $\Sigma \not\models \sigma$. It follows that $\mathcal{N} \not\models Prv_\Sigma(\underline{\sharp\sigma})$, so $\mathcal{N} \models \neg Prv_\Sigma(\underline{\sharp\sigma})$, and hence $\mathcal{N} \models \sigma$. Therefore, $\sigma \in Th(\mathcal{N}) \backslash Cn(\Sigma)$, so $Cn(\Sigma) \neq Th(\mathcal{N})$.

It follows that $Th(\mathcal{N})$ is not axiomatizable. □

Appendix A

Mathematical Background

A.1 Terminology and Notation

Definition A.1.1. *We let* $\mathbb{N} = \{0, 1, 2, \dots\}$ *and we let* $\mathbb{N}^+ = \mathbb{N}\backslash\{0\}$.

Definition A.1.2. *For each* $n \in \mathbb{N}$*, we let* $[n] = \{m \in \mathbb{N} : m < n\}$*, so* $[n] = \{0, 1, 2, \dots, n-1\}$.

We will often find a need to work with finite sequences, so we establish notation here.

Definition A.1.3. *Let* X *be a set. Given* $n \in \mathbb{N}$*, we call a function* $\sigma\colon [n] \to X$ *a* finite sequence from X of length n*. We denote the set of all finite sequences from* X *of length* n *by* X^n*. We use* λ *to denote the unique sequence of length* 0*, so* $X^0 = \{\lambda\}$*. Finally, given a finite sequence* σ *from* X*, we use the notation* $|\sigma|$ *to denote the length of* σ*.*

Definition A.1.4. *Let* X *be a set. We let* $X^* = \bigcup_{n\in\mathbb{N}} X^n$*, i.e.* X^* *is the set of all finite sequences from* X*.*

We denote finite sequences by simply listing the elements in order. For instance, if $X = \{a, b\}$, the sequence *aababbba* is an element of X^*. Sometimes for clarity, we'll insert commas and instead write a, a, b, a, b, b, b, a.

Definition A.1.5. *If* $\sigma, \tau \in X^*$*, we denote the concatenation of* σ *and* τ *by* $\sigma\tau$ *or* $\sigma * \tau$*.*

Definition A.1.6. *If* $\sigma, \tau \in X^*$*, we say that* σ *is an* initial segment *of* τ*, a write* $\sigma \preceq \tau$*, if* $\sigma = \tau \restriction [n]$ *for some* n*. We say that* σ *is a* proper initial segment *of* τ*, and write* $\sigma \prec \tau$ *if* $\sigma \preceq \tau$ *and* $\sigma \neq \tau$*.*

Definition A.1.7. *Given a set* A*, we let* $\mathcal{P}(A)$ *be the set of all subsets of* A*, and we call* $\mathcal{P}(A)$ *the* power set *of* A*.*

For example, we have $\mathcal{P}(\{1, 2\}) = \{\emptyset, \{1\}, \{2\}, \{1, 2\}\}$ and $\mathcal{P}(\emptyset) = \{\emptyset\}$. A simple combinatorial argument shows that if $|A| = n$, then $|\mathcal{P}(A)| = 2^n$.

A.2 Countable Sets

Definition A.2.1. *Let* A *be a set.*

- *We say that* A *is* countably infinite *if there exists a bijection* $f\colon \mathbb{N} \to A$*.*
- *We say that* A *is* countable *if it is either finite or countably infinite.*

Proposition A.2.2. *Let* A *be a nonempty set. The following are equivalent:*

1. *A is countable.*
2. *There exists a surjection $g\colon \mathbb{N} \to A$.*
3. *There exists an injection $h\colon A \to \mathbb{N}$.*

Proposition A.2.3. *We have the following:*

1. *If A and B are both countable, then $A \times B$ is countable.*
2. *If $A_0, A_1, A_2, \ldots$ are all countable, then $\bigcup_{n\in\mathbb{N}} A_n$ is countable.*

Corollary A.2.4. *If A is countable, then A^* is countable.*

Theorem A.2.5. *The sets $\mathbb{Z}$ and $\mathbb{Q}$ are countably infinite, but $\mathbb{R}$ and $\mathcal{P}(\mathbb{N})$ are not countable.*

A.3 Partial and Linear Orderings

Definition A.3.1. *Let A be a set.*

1. *A* weak partial ordering *on A is a binary relation $\leq$ on A with the following properties:*
   - *$a \leq a$ for all $a \in A$.*
   - *If $a \leq b$ and $b \leq a$, then $a = b$.*
   - *If $a \leq b$ and $b \leq c$, then $a \leq c$.*
2. *A* weak linear ordering *on A is a weak partial ordering $\leq$ on A such that for all $a, b \in A$, either $a \leq b$ or $b \leq a$.*
3. *A* strict partial ordering *on A is a relation $<$ on A with the following properties:*
   - *$a \not< a$ for all $a \in A$.*
   - *If $a < b$ and $b < c$, then $a < c$.*
4. *A* strict linear ordering *on A is a strict partial ordering $<$ on A such that for all $a, b \in A$, at least of one $a = b$, $a < b$ or $b > a$ is true.*

Proposition A.3.2. *If $<$ is a strict partial ordering on A, and $a < b$, then $b \not< a$.*

Proof. Suppose that $a < b$. If we also have $b < a$, then by transitivity, it would follow that $a < a$, contradicting the first condition. Therefore, we must have $b \not< a$. □

Proposition A.3.3. *Suppose that $\leq$ is a weak partial ordering on A. Define a relation $<$ on A by saying that $a < b$ if both $a \leq b$ and $a \neq b$. We then have that $<$ is a strict partial ordering on A. Moreover, if $\leq$ is a weak linear ordering, then $<$ is a strict linear ordering.*

Proof. We first check the three properties to show that $<$ is a strict partial ordering:

- For any $a \in A$, we trivially have $a = a$, so $a \not< a$ by definition.
- Let $a, b \in A$ be arbitrary with $a < b$. By definition, we then have that $a \leq b$ and $a \neq b$. Now if $b < a$, then $b \leq a$ by definition, and combining this with $a \leq b$ we conclude that $a = b$, contradicting the fact that $a < b$. Thus, we must have $b \not< a$.

- Let $a, b, c \in A$ be arbitrary such that both $a < b$ and $b < c$. We then have both $a \leq b$ and $b \leq c$, so we conclude that $a \leq c$. Notice that $a \neq c$ by what we just proved (if $a = c$, then we would have both $a < b$ and $b < a$). Therefore, $a < c$.

Thus, $<$ is a strict partial ordering on A.

Finally, suppose that $\leq$ is a weak linear ordering. Let $a, b \in A$ be arbitrary. If $a = b$, then we are done, so assume that $a \neq b$. We know by assumption that either $a \leq b$ or $b \leq a$. In the former case, we have $a < b$ by definition, and in the latter we have $b < a$. Therefore, at least of one $a = b$, $a < b$ or $b > a$ is true. □

Proposition A.3.4. *Suppose that $<$ is a strict partial ordering on A. Define a relation $\leq$ on A by saying that $a \leq b$ if either $a < b$ or $a = b$. We then have that $\leq$ is a weak partial ordering on A. Moreover, if $<$ is a strict linear ordering, then $\leq$ is a weak linear ordering.*

Proof. We first check the three properties to show that $\leq$ is a weak partial ordering:

- For any $a \in A$, we trivially have $a = a$, so $a \leq a$ by definition.
- Let $a, b \in A$ be arbitrary such that both $a \leq b$ and $b \leq a$. Suppose for the sake of obtaining a contradiction that $a \neq b$. Since $a \leq b$, we conclude by definition $a < b$. Similarly, we have $b < a$. This is a contradiction to the fact that $<$ is a strict partial ordering, so we must have $a = b$.
- Let $a, b, c \in A$ be arbitrary such that both $a \leq b$ and $b \leq c$. If $a = b$, then clearly $a \leq c$. Similarly, if $b = c$, then clearly $a \leq c$. Suppose then that both $a \neq b$ and $a \neq c$. We have $a < b$ and $b < c$ by definition, so $a < c$, and hence $a \leq c$.

Thus, $\leq$ is a weak partial ordering on A.

Finally, suppose that $<$ is a strict linear ordering. Let $a, b \in A$ be arbitrary. We know by assumption that at least of one $a = b$, $a < b$ or $b > a$ is true. In the firs two cases, we clearly have $a \leq b$, while in the last case we have $b \leq a$. □

A.4 Ordered Groups, Rings, and Fields

Definition A.4.1. *Let $(A, +, 0)$ be an abelian group.*

1. *Given a weak linear ordering $\leq$ on A, we say that $\leq$ is compatible with the group structure if whenever $a, b, c \in A$ and $a \leq b$, we have $a + c \leq b + c$.*
2. *Given a strict linear ordering $<$ on A, we say that $<$ is compatible with the group structure if whenever $a, b, c \in A$ and $a < b$, we have $a + c < b + c$.*

Proposition A.4.2. *Let $(A, +, 0)$ be an abelian group.*

1. *If $\leq$ is a weak linear ordering compatible with the group structure, then $<$ is a strict linear ordering compatible with the group structure.*
2. *If $<$ is a strict linear ordering compatible with the group structure, then $\leq$ is a weak linear ordering compatible with the group structure.*

Proof.

1. Suppose that $\leq$ is a weak linear ordering that is compatible with the group structure. By Proposition A.3.3, it follows that $<$ is a strict linear ordering. Now let $a, b, c \in A$ be arbitrary with $a < b$. Since $a < b$, we know that $a \leq b$, and hence $a + c \leq b + c$. Now if $a + c = b + c$, then by adding $-c$ to both sides we would have $a = b$, which is a contradiction. Therefore, $a + c < b + c$.

2. Suppose that $<$ is a strict linear ordering that is compatible with the group structure. By Proposition A.3.4, it follows that $\leq$ is a weak linear ordering. Now let $a, b, c \in A$ be arbitrary with $a \leq b$. We have two cases:

   - *Case 1:* Suppose that $a = b$. In this case, we immediately conclude that $a + c = b + c$, and hence $a + c \leq b + c$.
   - *Case 2:* Suppose that $a < b$. Since $<$ is compatible with the group structure, it follows that $a + c < b + c$, and hence $a + c \leq b + c$.

   Therefore, $a + c \leq b + c$ in either case.

□

Definition A.4.3. *An* ordered abelian group *is an abelian group $(A, +, 0)$ together with a linear ordering (either weak or strict) that is compatible with the group structure.*

Proposition A.4.4. *Let $(A, +, 0, \leq)$ be an ordered abelian group. If $a \leq b$ and $c \leq d$, then $a + c \leq b + d$.*

Proof. Let $a, b, c, d \in A$ be arbitrary with $a \leq b$ and $c \leq d$. Since $a \leq b$ and $c \in A$, we know that $a+c \leq b+c$. Similarly, since $c \leq d$ and $b \in A$, we have $c+b \leq d+b$. Using the fact that $+$ is commutative, it follows that $b + c \leq b + d$. Finally, since we have both $a + c \leq b + c$ and also $b + c \leq b + d$, we can use the transitivity of $\leq$ to conclude that $a + c \leq b + d$. □

Definition A.4.5. *An* ordered (commutative) ring *is a ring R (with identity) together with a binary relation $\leq$ such that*

- *$\leq$ is a weak linear ordering of R.*
- *If $a \leq b$ and $c \in R$, then $a + c \leq b + c$.*
- *If $0 \leq a$ and $0 \leq b$, then $0 \leq ab$.*

Definition A.4.6. *An* ordered (commutative) ring *is a ring R (with identity) together with a binary relation $\leq$ such that*

- *$<$ is a strict linear ordering of R.*
- *If $a < b$ and $c \in R$, then $a + c < b + c$.*
- *If $0 < a$ and $0 < b$, then either $0 < ab$ or $ab = 0$.*

As above, it's easy to see that given either of these definitions, if we define the other relation as above, then it satisfies the properties of the other. Notice that we need to add the possibility that $ab = 0$ in the latter case. IS THERE AN EXAMPLE? WHAT ABOUT $\mathbb{Z} \times \mathbb{Z}$ with lexicographic - No $(0, 1) \cdot (1, -1) = (0, -1)$? If R is integral domain, however, then of course we can drop that.

Proposition A.4.7. *Let R be an ordered ring and let $a, b \in R$.*

- *$a \leq b$ if and only if $0 \leq b - a$.*
- *$a < b$ if and only if $0 < b - a$.*

Proof. If $a \leq b$, then $a + (-a) \leq b + (-a)$, so $0 \leq b - a$. Conversely if $0 \leq b - a$, then $a \leq (b - a) + a$, so $a \leq b$. The same arguments work for the strict inequality. □

Proposition A.4.8. *Let R be an ordered ring and let $a, b \in R$.*

- *$a \leq b$ if and only if $-b \leq -a$.*
- *$a < b$ if and only if $-b < -a$.*

Proof. Suppose first that $a < b$. We then have that $a + ((-a) + (-b)) < b + ((-a) + (-b))$, hence $-b < -a$. Suppose conversely that $-b < -a$. We then have that $-(-a) < -(-b)$ by what we just proved, hence $a < b$ (here we are using the ring theory fact that $-(-a) = a$ for all a). The same arguments work for the strict inequality. □

Proposition A.4.9. *Let R be an ordered ring and let $a, b, c \in R$.*

- *If $a \leq b$ and $0 \leq c$, then $ac \leq bc$.*
- *If $a \leq b$ and $c \leq 0$, then $bc \leq ac$.*
- *If $a < b$ and $0 < c$ and c is not a zero divisor, then $ac < bc$.*
- *If $a < b$ and $c < 0$ and c is not a zero divisor, then $bc < ac$.*

Proof. Suppose $0 \leq c$. Since $a \leq b$, we have $0 \leq b - a$, so $0 \leq (b - a)c$, so $0 \leq bc - ac$, so $ac \leq bc$.

Suppose now that $c \leq 0$. We then have $0 \leq -c$, so by what we just proved it follows that $a(-c) \leq b(-c)$, or $-ac \leq -bc$. The previous proposition then gives $bc \leq ac$. □

Proposition A.4.10. *Let R be an ordered ring and let $a, b \in R$.*

- *If $a \geq 0$ and $b \geq 0$, then $ab \geq 0$.*
- *If $a \geq 0$ and $b \leq 0$, then $ab \leq 0$.*
- *If $a \leq 0$ and $b \geq 0$, then $ab \leq 0$.*
- *If $a \leq 0$ and $b \leq 0$, then $ab \geq 0$.*

Proof. The first is an axiom. Suppose that $a \geq 0$ and $b \leq 0$. From above, we know that $-0 \leq -b$, i.e. that $0 \leq -b$. Using the first, we conclude that $0 \leq a(-b)$, so $0 \leq -ab$. From above, it follows that $ab \leq 0$. The third follows similarly (or from the second and commutativity).

Suppose now that $a \leq 0$ and $b \leq 0$. We then have $0 \leq -a$ and $0 \leq -b$, so $0 \leq (-a)(-b)$, from which we conclude that $0 \leq ab$. □

Corollary A.4.11. *Let R be an ordered ring. We have $0 \leq a^2$ for all $a \in R$.*

Proof. Either $0 \leq a$ or $a \leq 0$, and each case now follows from the previous proposition. □

Corollary A.4.12. *If R is an ordered ring, then $0 \leq 1$ and $-1 \leq 0$. Furthermore, if $0 \neq 1$, then $0 < 1$ and $-1 < 0$.*

Proof. This is immediate from $1^2 = 1$ and the above. □

Proposition A.4.13. *Let R be an ordered ring and let G be the set of nonnegative elements of R, i.e. $G = \{a \in R : a \geq 0\}$.*

- *If $a, b \in G$, then $a + b \in G$.*
- *If $a, b \in G$, then $ab \in G$.*
- *If $a \in G$ and $a \neq 0$, then $\frac{1}{a} \in G$.*

Proof. Suppose that $a, b \in G$. We have $0 \leq a$, hence $0 + b \leq a + b$, or $b \leq a + b$. Since we also have $0 \leq b$, we can use transitivity of $\leq$ to conclude that $0 \leq a + b$. Thus, $a + b \in G$. Trivially we have $ab \in G$ using the last assumption.

Suppose now that $a \in G$ and $a \neq 0$. We then have $a > 0$. Notice that $a \cdot a^{-1} = 1 > 0$. If $a^{-1} \leq 0$, then since $a \geq 0$, we would have $1 = a \cdot a^{-1} \leq 0$, a contradiction (unless $1 = 0$, in which case this is trivially since $G = R$). □

A.5 Lattices and Boolean Algebras

Lattices as special (weak) partial orderings. Every pair of elements has a greatest lower bound and least upper bound.

A.6 Algebraically Closed Fields